MASTER EN SCIENCES MATHÉMATIQUES

Comparaison entre le gap test et le test des hypervolumes en classification

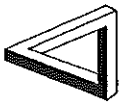
Beauthier, Catherine

Award date:
2002

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Comparaison entre le gap test et le test des hypervolumes en classification



Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

Promoteur : A. Hardy

Catherine BEAUTHIER

Année académique 2001-2002

Je remercie tout d'abord Monsieur Hardy, promoteur de ce mémoire, de m'avoir guidée tout au long de la réalisation de ce travail.

Je remercie également ma famille, mes amis et toutes les personnes qui m'ont aidée et soutenue durant mes années d'université.

Résumé

Un des problèmes liés à la classification automatique est de déterminer le nombre de classes naturelles présentes dans les données. Dans ce travail, nous présentons et implémentons le gap test une méthode de détermination du nombre de classes basée sur le critère des hypervolumes. Ensuite, après une brève présentation des données symboliques, nous comparons les résultats donnés par le gap test et une autre méthode de détermination du nombre de classes basée sur le critère des hypervolumes, le test des hypervolumes. Cette comparaison est faite sur base d'ensembles de données classiques, mais également sur des données symboliques de type intervalle.

Abstract

One of the classification problems is the determination of the true number of clusters in the data. In this work, we introduce and implement a method used to determine the number of clusters based on the hypervolumes criterion named the gap test. We introduce too the symbolic data and compare the results obtained by the application of the gap test and the hypervolume test, an other method based on the hypervolumes criterion to classical data sets and symbolic data of interval type.

Table des matières

Introduction 4

1 La classification automatique 6
- 1.1 Le problème . . . 6
- 1.2 Complexité du problème de classification . . . 8
- 1.3 Classification et dissection . . . 9
- 1.4 Les étapes de la classification automatique . . . 10
  - 1.4.1 La collecte des données . . . 10
  - 1.4.2 Le calcul des proximités . . . 11
  - 1.4.3 Les stratégies de classification . . . 12
- 1.5 Les méthodes hiérarchiques utilisées . . . 13
  - 1.5.1 La méthode du lien simple . . . 14
  - 1.5.2 La méthode du lien complet . . . 14
  - 1.5.3 La méthode du centroïde . . . 15
  - 1.5.4 La méthode de Ward . . . 15
- 1.6 La méthode des hypervolumes [4] . . . 16
  - 1.6.1 Le modèle . . . 16
  - 1.6.2 Le processus de Poisson . . . 17
  - 1.6.3 Le critère . . . 18

2 La détermination du nombre de classes 21
- 2.1 Introduction . . . 21
- 2.2 La notion de classe naturelle . . . 21
- 2.3 Méthodes de détermination du nombre de classes basées sur le critère des hypervolumes [5] . . . 23
  - 2.3.1 Méthode géométrique classique, la méthode du coude . 23
  - 2.3.2 Méthode basée sur l'estimation d'un ensemble convexe . . . 23
  - 2.3.3 Test du quotient de vraisemblance . . . 25

2.4 Le gap test [7] 27
2.4.1 Le test 28
2.4.2 Le quotient de vraisemblance 29
2.4.3 La région critique 30
2.4.4 Distribution de la statistique du test 30

3 Comparaison des résultats pour des données classiques **33**
3.1 Introduction 33
3.2 Présentation des programmes 33
3.2.1 Programme du gap test 33
3.2.2 Programme du test des hypervolumes 34
3.3 Présentation des résultats 34
3.3.1 Deux groupes, de tailles inégales en dimension 2 34
3.3.2 Deux groupes parallèles en dimension 2 38
3.3.3 Trois classes parallèles en dimension 3 43
3.3.4 Trois classes parallèles en dimension 2 43
3.3.5 Deux groupes éloignés en dimension 3 47
3.3.6 Deux groupes proches en dimension 3 49
3.3.7 Trois groupes bien séparés en dimension 2 51
3.3.8 Deux classes bien séparées en dimension 2 52
3.3.9 Trois classes allongées en dimension 2 53
3.3.10 Données non séparables en dimension 3 57
3.3.11 Données sans structure en dimension 2 58
3.3.12 Données de Ruspini, quatre classes en dimension 2 59
3.3.13 Deux classes non convexes en dimension 2 64
3.3.14 Quatre classes non convexes en dimension 3 68
3.4 Conclusion 72

4 Les données symboliques **73**
4.1 Introduction 73
4.2 Les données classiques 74
4.2.1 Les variables quantitatives 74
4.2.2 Les variables qualitatives 74
4.2.3 Les variables dépendantes 75
4.3 Les données symboliques 75
4.3.1 Quelques exemples de données symboliques 76
4.3.2 Variables multivaluées et intervalles 77
4.3.3 Variables multivaluées et intervalles par agrégation . . 78

4.3.4 Variable modale 79
4.3.5 Tableau de données symboliques 81
4.4 Les objets symboliques 82
4.5 La détermination du nombre de classes pour des données symboliques de type intervalle 83

5 Résultats du gap test et du test des hypervolumes appliqués à des données symboliques de type intervalle **86**
5.1 Introduction 86
5.2 Données symboliques de type intervalle, deux variables 86
5.3 Données symboliques de type intervalle, quatre variables . . . 90

Conclusion **93**

Bibliographie **97**

Introduction

Beaucoup de domaines d'activités ont aujourd'hui recours à l'analyse des données pour extraire les informations importantes de leurs travaux.

La classification automatique fait partie de l'analyse des données et possède de nombreuses applications dans beaucoup de domaines. Elle permet de classer des objets décrits par un ensemble de caractéristiques en un certain nombre de sous-groupes d'objets semblables appelés classes. Il existe aujourd'hui énormément de méthodes de classification.

Un des problèmes de la classification est la détermination du nombre de classes. En effet, la plupart des méthodes de classification ne permettent pas de déterminer ce nombre. Elles fournissent une partition des données en un certain nombre de classes où ce nombre est fixé par l'utilisateur. Or celui-ci ne connait pas toujours a priori le nombre de groupes naturels présents dans les données. C'est pourquoi de nombreuses techniques de détermination du nombre de classes ont été proposées. Elles permettent de déterminer le nombre de classes naturelles présentes dans les données sur base de partitions obtenues par des méthodes de classification. Deux de ces méthodes sont le gap test et le test des hypervolumes, celui-ci étant basé sur la méthode de classification des hypervolumes.

L'objectif de ce mémoire est de présenter le gap test et de l'implémenter afin de pouvoir l'appliquer à plusieurs jeux de données. Nous avons ensuite comparé les résultats obtenus avec ceux de l'application du test des hypervolumes à ces mêmes jeux de données.

De plus, le monde réel est trop complexe pour être décrit par des variables classiques. C'est pourquoi on utilise aujourd'hui les variables symboliques. Dans ce travail, nous avons également appliqué le gap test et le test des hypervolumes à des jeux de données symboliques afin de comparer les résultats obtenus.

Nous rappellerons tout d'abord dans le premier chapitre en quoi consiste la classification automatique. Nous présenterons également les méthodes de classification que nous appliquerons aux différents jeux de données à traiter.

Nous présenterons ensuite dans un deuxième chapitre, les deux méthodes de détermination du nombre de classes que nous allons considérer, le gap test et le test des hypervolumes.

Dans le troisième chapitre de ce travail, nous détaillerons les résultats obtenus par l'application du gap test et du test des hypervolumes à des jeux de données classiques.

Dans le chapitre suivant nous présenterons les données symboliques et dans le dernier chapitre, les résultats obtenus par l'application des deux tests à des jeux de données symboliques de type intervalle.

Chapitre 1

La classification automatique

Dans ce chapitre, nous détaillerons en quoi consiste un problème de classification et nous présenterons les différentes méthodes de classification utilisées par la suite dans ce travail.

1.1 Le problème

L'objectif premier de la classification automatique est de proposer une solution au problème suivant :

"Comment décomposer une population donnée, d'individus ou d'objets, décrits par un ensemble de caractéristiques, en un certain nombre de groupes homogènes appelés classes."

De nombreux chercheurs ont étudié ce problème et beaucoup d'applications ont vu le jour dans de nombreux domaines comme la biologie, la chimie, la médecine, la linguistique, ... En biologie par exemple, on essaie de regrouper des animaux ou des plantes qui possèdent les mêmes caractéristiques. Cette discipline est appelée la taxonomie.

Les techniques de classification automatique permettent donc de réduire de grands ensembles de données multidimensionnelles à un nombre restreint de sous-groupes appelés classes.

L'apparition des ordinateurs a permis une résolution plus rapide des problèmes de classification et de leurs nombreux calculs, ce qui a provoqué l'introduction de nouveaux concepts et par la suite de nouveaux algorithmes

de plus en plus diversifiés, performants et rigoureux. Les domaines d'application de la classification sont aujourd'hui de plus en plus nombreux et les techniques pour le résoudre tout aussi diverses.

"Parallèlement, de mêmes idées, de mêmes algorithmes, sont publiés de façons indépendantes et souvent sous des noms différents. Face à cette littérature si riche et si diversifiée, l'utilisateur potentiel de la classification n'a pas la possibilité ni le temps d'en faire la synthèse et ne peut donc choisir la méthode la plus appropriée à son problème." (Chandon et Pinson, 1981 [3])

De manière mathématique, on peut définir un problème de classification automatique de la façon suivante:

Soit

- $E = \{x_1, \ldots, x_n\}$: l'ensemble des individus à classifier. On suppose que $\#E < \infty$;

- $V = \{v_1, v_2, \ldots, v_p\}$: l'ensemble des variables que l'on mesure sur chaque individu.

L'objectif est de répartir les individus en classes de sorte que les individus d'une même classe possèdent des caractéristiques qui les distinguent des membres des autres classes. Par la suite on essaie aussi de trouver les caractéristiques communes de chaque classe afin de les étiqueter.

Remarquons que pour des valeurs de p égales à 1, à 2 ou à 3, on peut facilement représenter les individus sur une droite, dans le plan ou dans l'espace. Il nous est alors possible de déterminer la présence de classes à l'oeil nu. Pour des valeurs de p plus élevées cela n'est plus envisageable.

Pour notre problème on recherche donc une partition de E en k classes (k fixé)

$$P = \{C_1, \ldots, C_k\}.$$

Pour cela, la plupart des méthodes de classification se basent sur un critère qui mesure la qualité de la partition P.

Soit $\mathcal{P}_k$ l'ensemble de toutes les partitions de E en k classes. Le critère de classification associé à chaque $P \in \mathcal{P}_k$ est:

$$\begin{array}{rccl} W: & \mathcal{P}_k & \rightarrow & \mathbb{R} \\ & P & \rightsquigarrow & W(P,k). \end{array}$$

Le problème de classification revient donc à trouver la partition optimale $P^* = \{C_1^*, C_2^*, \ldots, C_k^*\}$ qui minimise ou maximise la valeur du critère, i.e. telle que:

$$W(P^*,k) = \min_{P \in \mathcal{P}_k} W(P,k)$$

ou

$$W(P^*,k) = \max_{P \in \mathcal{P}_k} W(P,k)$$

Les différentes méthodes de classification se distinguent par le critère utilisé. Pour la plupart des méthodes, le critère est basé sur une mesure de distance ou de dissimilarité comme nous le verrons par la suite.

1.2 Complexité du problème de classification

La difficulté du problème de classification tel que nous venons de le décrire, vient de sa nature combinatoire. En effet, comme l'ensemble des données E est fini, une solution au problème de classification serait de construire toutes les partitions possibles en k classes de l'ensemble E et ensuite choisir la meilleure partition au sens du critère.

En pratique cependant, nous ne choisirons pas cette solution. En effet, le nombre de partitions de n objets en k classes est donné par le nombre de Stirling d'ordre 2:

$$S(n,k) = \frac{1}{k!} \sum_{i=1}^{n} C_n^i \, (-1)^{k-i} \, i^n$$

où

$$\begin{array}{rcl} S(n,k) & = & k\, S(n-1,k) + S(n-1,k-1); \\ S(1,1) & = & 1; \\ S(1,n) & = & 0 \text{ (pour } n \neq 1). \end{array}$$

On aura par exemple,

- $S(n,2) = 2^{n-1} - 1$,
- $S(59,2) = 10^{18}$,
- $S(15,3) = 2\ 375\ 101$
- $S(25,10) = 1\ 203\ 163\ 392\ 175$,
- $S(100,5) = 10^{68}$

Si un ordinateur pouvait évaluer un million de partitions par seconde, il lui faudrait plus de 8 jours pour déterminer la partition optimale de 20 objets en 5 classes et plus de 2444 siècles si le nombre d'objets passe à 30. C'est pourquoi un algorithme passant en revue toutes les partitions de n individus en k classes ne pourra être implémenté que pour de petites valeurs de n et k.

Pour résoudre ce problème, il existe des procédures de programmation dynamiques qui éliminent les calculs inutiles d'une énumération complète et qui convergent néanmoins vers une solution optimale. Des algorithmes de type "Branch and bound" peuvent également être utilisés. Il existe aussi des approches heuristiques intéressantes.

1.3 Classification et dissection

Il est important de faire remarquer que la classification est différente de la dissection. Rappelons en effet que le premier objectif de la classification est d'examiner la structure d'un ensemble d'individus et plus particulièrement de déterminer si ceux-ci se répartissent naturellement en un nombre de groupes d'objets appelés classes. On ne suppose pas cependant que les objets se répartissent a priori en classes. La conclusion d'une telle étude peut être qu'il n'y a pas de classes dans les données, mais si des groupes distincts existent, ils doivent être découverts. Le second objectif est l'analyse des résultats obtenus et éventuellement l'émission de nouvelles hypothèses sur le phénomène étudié.

Par contre, dans un problème de dissection, on est en présence d'un seul groupe d'objets et le but est de diviser ce groupe en un nombre fixé de sous-groupes ayant des propriétés communes. Par exemple, on peut vouloir diviser un groupe de maisons en cinq districts postaux d'effectifs comparables. Dans

ce cas, deux maisons de districts différents peuvent être plus semblables que deux maisons d'un même district.

1.4 Les étapes de la classification automatique

La première étape est la collecte des données. Le choix des individus et des variables aura bien sûr une grande influence sur les résultats de l'analyse.

La deuxième étape concerne le calcul des dissimilarités entre toutes les paires d'objets. En effet, la classification automatique a pour objectif la division d'un ensemble d'individus en sous-groupes d'individus semblables. Le choix d'une mesure de dissimilarité dépendra entre autre de la nature du problème et du choix des individus et des variables.

La troisième étape est le choix de l'algorithme. Ce choix dépendra du type et de la structure de classe que l'on veut obtenir, du critère défini et de la stratégie de classification que l'on veut adopter.

Une procédure de classification est avantageuse si elle est accompagnée d'un test qui permet de déterminer la présence ou l'absence d'une structure dans les données. De plus, une classification stable donnera plus de poids aux résultats. Ceci constitue la quatrième étape.

La dernière étape est l'interprétation des résultats. Un examen minutieux de chacune des classes obtenues permettra en général de confirmer ou de réfuter une hypothèse émise a priori.

Remarquons que toutes les procédures de classification ne suivent pas les cinq étapes énumérées. Dans certains cas par exemple, les données se réduisent à une matrice de dissimilarités; certaines méthodes utilisent un critère qui ne se base pas sur une mesure de dissimilarité entre couples d'individus; d'autres méthodes n'utilisent pas explicitement de critère de classification.

1.4.1 La collecte des données

Avant toute chose dans l'étude d'une population, il faut choisir les individus et les variables afin d'obtenir des informations pertinentes. En général, la population est trop grande pour mesurer la valeur des variables sur tous ses individus. C'est pourquoi par un processus d'échantillonnage, on essaie

de déterminer un échantillon représentatif de la population.

Le choix des variables influence sans doute le plus les résultats finaux d'une analyse de classification. On s'efforce donc de choisir des variables qui caractériseront au mieux les individus et qui permettront de rassembler un maximum d'informations.

Remarquons qu'on peut rencontrer certaines difficultés lorsque les variables sont corrélées entre elles. Les résultats de l'étude peuvent être biaisés. Des solutions existent, notamment celle qui consiste à transformer les données de sorte que les nouvelles variables ne soient plus corrélées. Cette transformation est basée sur l'analyse factorielle en composantes principales qui permet aussi de réduire la dimension de l'espace des individus par projection. Il faut cependant rester prudent avec cette méthode car la structure éventuelle des données peut ne pas apparaître sur les graphes en dimensions réduites des points.

Une fois les individus et les variables déterminés, on peut recueillir les données et les représenter dans une matrice $(n \times p)$.

Ainsi, n individus sur lesquels on mesure p variables $(v_1, v_2, \ldots, v_p)$ peuvent être représentés dans une matrice M de la forme:

$$M = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

Dans la suite, nous supposerons que les variables sont quantitatives. Chaque individu pourra être représenté dans un espace euclidien à p dimensions.

1.4.2 Le calcul des proximités

Pour déterminer la présence de classes parmi les individus, la matrice des données ne suffit pas. La plupart des algorithmes de classification automatique déterminent une structure de classes en optimisant un critère défini à partir de dissimilarités mesurées entre toutes les paires d'objets à classifier. La classification sera bonne si ces dissimilarités mesurent correctement la proximité des individus. Le choix de l'indice de dissimilarité est donc important. La mesure de dissimilarité choisie devra être adaptée au type de variables utilisées (quantitatives, qualitatives, ...) et au type de phénomène

à étudier.

Remarquons que les critères de méthodes de classification ne sont pas tous basés sur une mesure de dissimilarité.

Une mesure de dissimilarité connue est la distance euclidienne:

$$d_{ij}^2 = \sum_{k=1}^{p} (x_{ik} - x_{jk})^2$$

Si on calcule l'indice choisi pour chaque paire d'individus, on peut construire la matrice des dissimilarités qui est une matrice symétrique:

$$D = \begin{pmatrix} d_{11} & \dots & d_{1p} \\ \vdots & \ddots & \vdots \\ d_{n1} & \dots & d_{np} \end{pmatrix}$$

1.4.3 Les stratégies de classification

La stratégie de classification que l'on va utiliser dépend du type de structure de classe que l'on veut obtenir (partitions, hiérarchies, ...) et du critère que l'on a choisi pour déterminer cette structure.

Le choix du critère est un problème important. Il détermine fortement la forme ou la taille des classes obtenues. Certains critères favorisent la formation de classes hypersphériques, d'autres conduisent à la détermination de classes allongées, ...

Remarquons que le critère des hypervolumes que nous verrons par la suite n'a pas été choisi mais déduit du modèle statistique sur lequel il est basé.

Parmi les nombreuses méthodes de classification, on peut distinguer:
- les méthodes monothétiques où la division d'un groupe en deux se fait sur base d'une seule variable;
- les méthodes polythétiques qui tiennent compte de toutes les variables simultanément.

On distingue aussi les méthodes hiérarchiques et non hiérarchiques. Les premières génèrent une hiérarchie de partitions et les algorithmes sont généralement de deux types, agglomératifs et divisifs.

Les algorithmes agglomératifs partent des n singletons et réunissent à chaque étape les deux classes les plus semblables au sens du critère choisi. Le processus s'arrête quand tous les individus sont rassemblés en une seule classe.

Les algorithmes divisifs partent de la classe contenant tous les individus et divisent en deux, à chaque étape, la classe contenant les groupes les plus dissemblables selon le critère. Le processus s'arrête quand on obtient la partition en n singletons.

Remarquons que la hiérarchie de partitions obtenue par une méthode agglomérative et une méthode divisive, toutes deux basées sur un même critère et appliquées à un même ensemble de données, peuvent être différentes. De plus, la hiérarchie obtenue peut dépendre de l'ordre de lecture des données. En général, on préférera utiliser des méthodes agglomératives qui demandent souvent moins de calculs pour le passage d'une itération à la suivante que les méthodes divisives.

Les procédures hiérarchiques procèdent de façon optimale à chaque étape mais la partition en k classes obtenue n'est en général, pas optimale. C'est en partie dû au fait que la plupart des méthodes hiérachiques sont de type séquentiel. L'affectation d'un objet à un groupe à la $i$ème itération n'est jamais remise en cause dans les étapes ultérieures.

Contrairement aux méthodes hiérachiques, les méthodes non hiérachiques déterminent une seule partition des n objets en k classes. Certaines de ces méthodes procèdent en choisissant une partition initiale en k classes et déplacent les objets d'un groupe dans un autre pour obtenir une meilleure partition au sens d'un critère donné. La partition initiale peut être soit tirée au hasard, soit le résultat d'une autre méthode de classification ou obtenue par tout autre moyen. Parmi ces algorithmes, on peut citer les algorithmes de transfert ou du type "nuées dynamiques".

1.5 Les méthodes hiérarchiques utilisées

Nous présentons dans cette section les différentes méthodes hiérarchiques de classification que nous utiliserons par la suite dans ce travail. On a vu que les méthodes hiérarchiques réunissent ou divisent à chaque étape les classes qui sont les plus semblables ou dissemblables au sens du critère. Les

méthodes que nous allons détailler se distinguent par le critère choisi.

1.5.1 La méthode du lien simple

Cette méthode est aussi appelée méthode du saut minimum ou méthode du voisin le plus proche. La distance entre deux classes C_i et C_j d'une partition est la plus petite distance séparant un point d'une classe et un point de l'autre.

$$D(C_i,C_j) = \min_{\substack{x \in C_i \\ y \in C_j}} d(x,y)$$

A chaque étape, on regroupe les deux classes qui sont les plus proches au sens de ce critère.

Un des inconvénients de cette méthode est que la hiérarchie obtenue peut dépendre de l'ordre de lecture des données. Si on trouve plusieurs fois la même plus petite distance entre deux groupes, la méthode regroupe les deux premiers groupes considérés; l'unicité de la partition n'est donc pas garantie.

La méthode est peu robuste. Si on perturbe un peu les données, on peut obtenir une partition totalement différente.

Cette méthode retrouve les classes bien séparées et a l'avantage de retrouver aussi les classes allongées grâce à l'effet de chaînage qui peut également être un désavantage car les classes séparées par un pont ne sont pas retrouvées.

1.5.2 La méthode du lien complet

Cette méthode est aussi appelée méthode du saut maximum ou méthode du voisin le plus éloigné.

La distance entre deux classes C_i et C_j d'une partition est la plus grande distance qui sépare un point d'une classe et un point de l'autre.

$$D(C_i,C_j) = \max_{\substack{x \in C_i \\ y \in C_j}} d(x,y)$$

Cette méthode est aussi peu robuste et la hiérarchie obtenue dépend aussi de l'ordre de lecture des données. Elle détecte les classes bien séparées

mais fonctionne mal avec des classes allongées. Elle a tendance à former des groupes hypersphériques.

1.5.3 La méthode du centroïde

La distance entre deux classes C_i et C_j est la distance entre leurs centroïdes, c'est-à-dire leurs centres de gravité.

Soit la classe C_l à n_l éléments, le centroïde de la classe l est défini par :

$$m^{(l)} = (m_1^{(l)}, \ldots, m_p^{(l)})$$

où

$$m_j^{(l)} = \frac{1}{n_l} \sum_{i=1}^{n_l} x_{ij}^{(l)}$$

c'est-à-dire la moyenne des observations relatives à la variable j pour le groupe l.

A chaque étape, on fusionne les deux groupes dont les centroïdes sont les plus proches. Cette méthode a aussi tendance à former des groupes hypersphériques.

1.5.4 La méthode de Ward

La distance entre deux classes C_i et C_j peut être mesurée par :

$$\Delta E^2_{C_i,C_j} = e^2_{C_i \cup C_j} - e^2_{C_i} - e^2_{C_j}$$

où e_l^2 est l'erreur associée à la classe l :

$$e_l^2 = \sum_{i=1}^{n_l} \sum_{j=1}^{p} (x_{ij}^{(l)} - m_j^{(l)})^2$$

$$= \sum_{i=1}^{n_l} \|x_i^{(l)} - m^{(l)}\|^2$$

On fusionne les deux classes pour lesquelles $\Delta E^2_{C_i,C_j}$ est minimum. Cette méthode tend également à former des groupes hypersphériques.

1.6 La méthode des hypervolumes [4]

La méthode des hypervolumes que nous allons décrire dans cette section n'a pas été appliquée dans ce travail. Nous avons cependant utilisé une procédure de détermination du nombre de classes basée sur son critère et c'est pourquoi nous la présentons ici.

Comme nous venons de le voir en décrivant quelques méthodes de classification, la plupart des critères sont basés sur des mesures de dissimilarités. Peu de procédures de classification sont basées sur un modèle statistique car contrairement aux autres disciplines de l'analyse des données, la classification automatique fait très rarement référence à la statistique mathématique.

Cependant, la méthode de classification automatique des hypervolumes repose sur un modèle statistique dont l'approche est basée sur la théorie des processus ponctuels et sur l'estimation par la technique du maximum de vraisemblance d'un domaine convexe compact.

1.6.1 Le modèle

Le modèle se base sur les hypothèses suivantes :

- Les points sont distribués dans k sous-domaines disjoints $D_1, D_2, \ldots, D_k$.
- Les variables aléatoires qui comptent le nombre de points dans des régions disjointes sont indépendantes.
- Le nombre moyen de points dans chaque région est proportionnel à la mesure de Lebesgue de cette région.

Le seul processus qui satisfait à ces conditions est le processus de Poisson homogène que nous définissons ci-après.

1.6.2 Le processus de Poisson

N est un processus de Poisson sur $E \subset \mathbb{R}^p$ ssi

a) $\forall A_1, A_2, \ldots, A_k \subset E$, disjoints 2 à 2, les variables aléatoires $N(A_1), \ldots, N(A_k)$ sont indépendantes.

$(N(A_i)$: nombre de points dans $A_i)$

b) $\forall A \subset E$ et $k \geq 0$,

$$P(N(A) = k) = e^{-\mu(A)} \quad \frac{(\mu(A))^k}{k!}$$

où μ est une mesure qui donne une masse finie à tout ensemble borné.

N est un processus de Poisson homogène (ou stationnaire) sur $E \subset \mathbb{R}^p$ ssi

a) N est un processus de Poisson sur E

b) $\mu(A) = \rho\, m(A)$

où $m(A)$ est la mesure de Lebesgue de A et $\rho \in \mathbb{R}_0^+$ est l'intensité du processus

Caractérisation du processus de Poisson homogène

On peut déduire d'un processus de Poisson N les deux caractéristiques suivantes :

1) $N(A)$, le nombre de points dans tout ensemble borné a une distribution de Poisson de moyenne $\rho\, m(A)$

2) Propriété d'uniformité conditionnelle: si $N(A) = n$, alors les n réalisations sont indépendantes et forment un échantillon aléatoire d'une loi **uniforme** sur A

1.6.3 Le critère

On considère l'ensemble $E = \{x_1, \ldots, x_n\}$ des individus à classifier, les points sont générés par un processus de Poisson homogène dans $D \in \mathbb{R}^p$ où

- $D = \bigcup_{i=1}^{k} D_i$, k est fixé
- les ensembles D_i sont disjoints et convexes

Le problème statistique est l'estimation des frontières des k ensembles $D_1, D_2, \ldots, D_k$. $C_i \subset \{x_1, x_2, \ldots, x_n\}$ est l'ensemble des observations appartenant à D_i $(1 \leq i \leq k)$.

Soit $\widetilde{x} = (x_1, x_2, \ldots, x_n)$, la réalisation du processus dans D.

Par la propriété d'uniformité conditionnelle du processus de Poisson, la fonction de vraisemblance s'écrit :

$$\begin{aligned} f(\widetilde{x}; D) &= f(x_1, \ldots, x_n; D) \\ &= \prod_{i=1}^{n} \frac{1}{m(D)} I_D(x_i) \\ &= \frac{1}{(m(D))^n} \prod_{i=1}^{n} I_D(x_i) \\ &= \frac{1}{(m(D))^n} I_D(H(\widetilde{x})) \end{aligned}$$

où $m(D)$ est la mesure de Lebesgue de D, $I_D(x_i)$ la fonction indicatrice de D et $H(\widetilde{x})$ est l'enveloppe convexe de $\widetilde{x}$.

Le problème est d'estimer le paramètre D, c'est-à-dire les k domaines D_i disjoints et convexes. Pour cela, on cherche l'estimateur du maximum de vraisemblance, c'est-à-dire l'estimateur des domaines D_i qui maximise la fonction de vraisemblance.

Or, le domaine D pour lequel la vraisemblance est maximale est, parmi ceux qui contiennent tous les points, celui dont la mesure de Lebesgue est minimale. Cependant si on n'impose pas de contraintes supplémentaires sur D, on constate qu'il y a moyen de trouver facilement k domaines D_i, qui

contiennent tous les points et tels que la somme de leurs mesures de Lebesgue soit nulle. (Fig. 1.1)

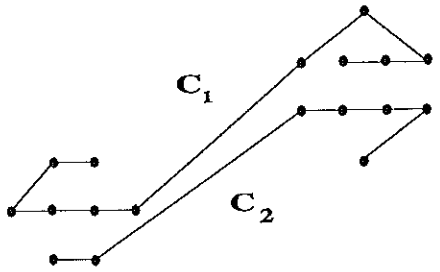


FIG. 1.1 –

Il y a beaucoup de solutions triviales à ce problème. On doit donc imposer une forme aux domaines D_i. On se limitera à l'hypothèse de convexité de ces domaines. (Ripley, Rasson (1977) [9])

En effet, on peut montrer que si D est convexe, l'enveloppe convexe de ses points est un estimateur du maximum de vraisemblance exhaustif pour D. Remarquons que cet estimateur est biaisé et qu'on peut résoudre ce problème en prenant comme estimateur de D la dilatation de l'enveloppe convexe de $\widetilde{x}$ à partir du centre de gravité des points de l'enveloppe convexe. Dans notre cas nous considérerons l'enveloppe convexe des points comme estimateur de D.

Ainsi, le maximum de la fonction de vraisemblance sera atteint par la partition pour laquelle la somme des mesures de Lebesgue des enveloppes convexes des k sous-domaines est minimale.

Le problème de classification et le critère des hypervolumes

L'estimation du maximum de vraisemblance nous permet de déterminer le critère suivant, on recherche une partition de E, $P = \{C_1, \ldots, C_k\}$ en k classes (k fixé). Avec $C_i \subset \{x_1, \ldots, x_n\}$ sous ensemble des points de D_i.

Le critère de classification est la somme des mesures de Lebesgue des enveloppes convexes des k classes.

$$\begin{aligned} \boldsymbol{W}: \ \mathcal{P}_k &\rightarrow \mathbb{R}^+ \\ \boldsymbol{P} &\rightsquigarrow \boldsymbol{W(P,k)} = \sum_{l=1}^{k} \boldsymbol{m(H(C_l))} \end{aligned}$$

où $\boldsymbol{m(H(C_l))}$ est la mesure de Lebesgue de l'enveloppe convexe des points appartenant à $\boldsymbol{C_l}$.

Le problème est donc de trouver $P^* \in \mathcal{P}_k$ tel que:

$$\begin{aligned} W(P^*,k) &= \min_{P \in \mathcal{P}_k} W(P,k) \\ &= \min \sum_{l=1}^{k} m(H(C_l)) \end{aligned}$$

Pratiquement, si le nombre de variables $p = 1$, la somme des mesures de Lebesgue des enveloppes convexes des classes sera la somme des longueurs des intervalles qui constituent les classes. Si $p = 2$, la somme des mesures de Lebesgue sera la somme des aires des enveloppes convexes. Ce critère généralise la mesure de Lebesgue sur $\mathbb{R}$ en la mesure de Lebesgue sur $\mathbb{R}^p$ (longueur sur la droite, surface dans le plan, ...) tandis que la plupart des procédures de classification automatique généralisent la notion de distance sur $\mathbb{R}$ en une distance ou une mesure de similarité sur $\mathbb{R}^p$.

On peut montrer aussi que le modèle qui vient d'être présenté peut s'interpréter comme un mélange de densités.

Si on applique la méthode des hypervolumes à plusieurs ensembles de données, on peut noter que la méthode n'a pas de problème avec les ponts entre les classes, qu'elle retrouve les classes allongées et qu'elle est robuste par rapport aux hypothèses de densité de points. Cependant, l'hypothèse importante est l'hypothèse de convexité des classes.

Chapitre 2

La détermination du nombre de classes

2.1 Introduction

Dans le problème de classification que nous avons présenté dans le chapitre précédent, on cherchait à déterminer une partition en k classes des données qui minimise ou maximise un critère. Cependant k est fixé, on ne sait pas ce qu'il vaut réellement. De plus, les quatre méthodes hiérachiques que nous avons présentées pour résoudre ce problème fournissent une répartition de l'ensemble des individus ou d'objets en groupes pour tous les nombres possibles de classes. Ces méthodes ne nous permettent pas de déterminer le nombre de classes naturelles présentes dans les données. C'est pourquoi il existe des méthodes permettant de déterminer ce nombre ou de vérifier l'absence de structure dans les données.

Avant de présenter, quelques méthodes de détermination du nombre de classes, nous allons préciser la notion de classe naturelle.

2.2 La notion de classe naturelle

Jusqu'à présent, nous avons utilisé plusieurs fois la notion de classe en évoquant le problème de classification et les méthodes utilisées pour le résoudre; nous ne l'avons cependant jamais définie.

Si on se base sur la définition du problème de classification, on peut caractériser une classe comme un ensemble d'individus ou d'objets ayant des

caractéristiques communes qui les distinguent des autres classes.

Si la mesure de similarité d'une paire d'objets est une distance, on peut alors définir la notion de classe comme étant un groupe de points tels que la distance entre deux points d'une même classe est plus petite que la distance entre un point d'une classe et un point d'une autre. Cependant, si on considère l'exemple ci-dessous (Fig. 2.1), on distingue clairement deux classes mais ces deux classes ne satisfont pas à la définition. En effet, on peut trouver deux points appartenant à chacune des classes qui sont plus proches que deux points appartenant à une même classe.

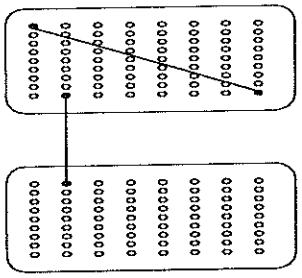


FIG. 2.1 –

En général, on parlera de cohésion interne et d'isolation externe pour la notion de classe. Une classe sera une région de l'espace contenant une densité relativement grande de points et les classes seront séparées par des régions de l'espace contenant peu de points.

Il est possible de mettre en évidence les qualités, les caractéristiques et les biais des méthodes de classification en les appliquant à des ensembles de données test dont on connait la structure en classes et en comparant la structure donnée par ces méthodes avec celle connue a priori.

Ainsi, dans le cas où nous sommes en présence de deux classes allongées parallèles, on peut dire que la méthode du lien simple est meilleure que

la méthode du lien complet. En effet, la méthode du lien simple retrouve les deux classes allongées tandis que la méthode du lien complet retrouve deux autres classes. On découvre ainsi que la méthode du lien complet a tendance à former des classes hypersphériques comme on l'a déjà remarqué précédemment. Cet exemple met aussi en évidence un problème plus général: certaines méthodes de classification imposent une structure particulière aux classes au lieu de trouver leur véritable structure. Or, la plupart des méthodes de détermination du nombre de classes se basent sur des suites de partitions en classes données par les méthodes de classification pour trouver le nombre optimal de classes.

2.3 Méthodes de détermination du nombre de classes basées sur le critère des hypervolumes [5]

Les méthodes que nous allons développer dans cette section sont basées sur le critère des hypervolumes vu précédemment. Rappelons que le critère des hypervolumes pour une partition P des données en k classes est donné par

$$W(P,k) = \sum_{l=1}^{k} m(H(C_l))$$

2.3.1 Méthode géométrique classique, la méthode du coude

Cette méthode consiste à tracer le graphe du critère $W(P,k)$ en fonction du nombre de classes k et de l'analyser. Le critère $W(P,k)$ décroît quand k croît. Une discontinuité dans la pente correspondra au nombre de classes naturelles présentes dans les données. Cette technique associée à la méthode des hypervolumes peut donner de bons résultats [5]; malheureusement, pour d'autres critères, on a montré que de grandes variations pouvaient apparaître dans la valeur du critère même s'il n'y a pas de structure dans les données.

2.3.2 Méthode basée sur l'estimation d'un ensemble convexe

Cette méthode est basée sur le problème suivant: «Etant donné la réalisation d'un processus de Poisson homogène dans un ensemble convexe D, estimer l'ensemble D en utilisant des méthodes d'inférence statistique.» La solution de ce problème est, comme on l'a déjà vu, la dilatation de l'enveloppe

convexe à partir de son centre de gravité. D est donc estimé par:

$$D' = g(H(D)) + c \cdot s(H(D))$$

où

- $H(D)$ est l'enveloppe convexe des points appartenant à D;
- $g(H(D))$ est le centroïde de $H(D)$;
- $s(H(D)) = H(D) - g(H(D))$.

La valeur de la constante c peut être estimée par $c = \sqrt{\frac{n}{n-v_n}}$ où v_n est le nombre de sommets de l'enveloppe convexe $H(D)$.

Notons que la réalisation d'un processus de Poisson dans l'union de k sous-ensembles $D_1, D_2, \ldots, D_k$ peut être considérée comme la réalisation de k processus de Poisson de même intensité dans les k sous-ensembles $D_1, D_2, \ldots, D_k$.

Notons par $P^* = \{C_1^k, C_2^k, \ldots, C_k^k\}$ la partition optimale de E en k classes et par $D_i'^k$ l'estimation de l'ensemble convexe D_i^k contenant C_i^k. Donc $D_i'^k$ est la version dilatée de D_i^k.

On propose la règle de décision suivante pour l'estimation de k:

- Si $D_1'^2 \bigcap D_2'^2 \neq \emptyset$, alors on conclut qu'il n'y a pas de structure dans les données.

- Si $\forall \{i,j\} \subset \{1,2,\ldots,t\}$, $i \neq j : D_i'^t \bigcap D_j'^t = \emptyset$, et que pour tout entier s tel que $2 \leq s < t$ et $\forall \{i,j\} \subset \{1,2,\ldots,s\}$, $i \neq j : D_i'^s \bigcap D_j'^s = \emptyset$, alors on conclut que la partition naturelle contient au moins t classes et on examine la partition en $(t+1)$ classes.

- S'il existe $\{i,j\} \subset \{1,2,\ldots,t\}$, $i \neq j : D_i'^t \bigcap D_j'^t \neq \emptyset$, et que pour tout entier s tel que $2 \leq s < t$ et $\forall \{i,j\} \subset \{1,2,\ldots,s\}$, $i \neq j : D_i'^s \bigcap D_j'^s = \emptyset$, alors on conclut que le nombre de classes naturelles k de l'ensemble des données est $t-1$.

Le défaut de cette méthode est le calcul des estimations des ensembles convexes et des intersections de ces ensembles. De plus, le coefficient de

dilatation diminue quand le nombre de points augmente, ce qui peut fausser les résultats. En effet, si on considère des données sans structure, il se peut qu'en augmentant le nombre de points, les dilatations des enveloppes convexes deviennent plus fines et ne possèdent plus d'intersection commune. La méthode conclura alors qu'il y a au moins deux classes dans les données bien qu'elle ait conclu en l'absence de structure avant l'augmentation du nombre de points.

2.3.3 Test du quotient de vraisemblance

La méthode des hypervolumes repose comme on l'a vu, sur un modèle statistique. On a supposé que les points observés résultaient de la réalisation d'un processus de Poisson homogène dans un ensemble D qui est l'union de k domaines convexes disjoints. Ce modèle statistique permet de formuler un test du quotient de vraisemblance pour la détermination du nombre de classes.

Soit $\{x_1,x_2,\ldots,x_n\}$ un échantillon aléatoire d'un processus de Poisson dans k ensembles convexes disjoints $D_1,D_2,\ldots,D_k$ dans un espace euclidien p-dimensionnel.

Pour un entier $k \geq 2$, nous testons si la subdivision en k classes est significativement meilleure que la subdivision en $k-1$ classes.

Nous testons donc,

H_0 : les données sont générées dans k domaines convexes disjoints $(t = k)$;

contre

H_A : les données sont générées dans $k-1$ domaines convexes disjoints. $(t = k-1)$.

Notons par :

- $C = \{C_1,C_2,\ldots,C_k\}$ la partition optimale de $\{x_1,x_2,\ldots,x_n\}$ en k classes.

- $D = \{D_1, D_2, \ldots, D_{k-1}\}$ la partition optimale de $\{x_1, x_2, \ldots, x_n\}$ en $k-1$ classes.

Comme nous l'avons vu précédemment, par la propriété d'uniformité conditionnelle du processus de Poisson, on peut écrire la fonction de vraisemblance pour le domaine D par exemple, comme ceci:

$$f_D(x) = \frac{1}{(m(D))^n} \prod_{i=1}^{n} I_D(x_i)$$

où $I_D(x_i)$ est la fonction indicatrice de l'ensemble D.

En se souvenant que les estimateurs du maximum de vraisemblance des domaines convexes C et D sont leur enveloppe convexe, le quotient de vraisemblance prend la forme de:

$$Q(x) = \frac{\sup_D f_D(x;\ t=k-1)}{\sup_C f_C(x;\ t=k)}$$

$$Q(x) = \frac{\frac{1}{\left(\sum_{i=1}^{k-1} m(H(D_i))\right)^n}}{\frac{1}{\left(\sum_{i=1}^{k} m(H(C_i))\right)^n}}$$

$$= \left(\frac{W(P,k)}{W(P,k-1)}\right)^n$$

$$= S^n$$

où $S = \frac{W(P,k)}{W(P,k-1)}$ et $W(P,k)$ est le critère des hypervolumes pour une partition en k classes.

La région critique sera de la forme:

$$\begin{aligned} RC &= \{Q(x) > c\} \\ &= \{\tfrac{W(P,k)}{W(P,k-1)} > c'\} \\ &= \{S > c'\} \end{aligned}$$

Puisque $W(P,k) \leq W(P,k-1)$ on a que $S \in [0,1]$ et on rejettera H_0 si S est grand. En effet, dans ce cas la valeur du critère des hypervolumes pour une partition en k classes ne sera pas significativement plus petite que la valeur du critère pour une partition en $k-1$ classes.

$$RC = \{(x_1,\ldots,x_n) : S \geq S_\alpha\}$$
$$\text{où } P_{H_0}(S \geq S_\alpha) = \alpha$$

Généralement, $\alpha = 0.01$ ou $\alpha = 0.05$. Le problème qui se pose est le suivant : on ne connaît pas la distribution de la statistique S. Nous ne pouvons donc pas examiner les propriétés théoriques du test ni déterminer le niveau de signification avec lequel on accepte ou rejette l'hypothèse H_0. En pratique cependant, on rejettera H_0 si S est proche de 1. Si k_0 est la première valeur de k pour laquelle on rejette H_0, alors on considère $k_0 - 1$ comme le nombre optimal de classes naturelles.

Ce test est appelé le **test des hypervolumes**.

2.4 Le gap test [7]

La méthode de détermination du nombre de classes que nous allons décrire ici est basée sur la théorie du processus de Poisson. Le gap test développe une règle d'arrêt exprimée en termes d'un test d'hypothèses.

Supposons que nous disposions d'une partition de E en v classes disjointes $\{D_1,D_2,\ldots,D_k,D_{k+1},\ldots,D_v\}$ $(v > k)$, où k est le nombre de classes naturelles présentes dans les données. Pour découvrir le nombre optimal k de classes, on propose de considérer un test qui utilise l'espace vide présent entre les différentes classes D_i, $i = 1,2,\ldots,v$.

Le test d'hypothèses que l'on propose permet de décider si les deux classes considérées doivent être réunies ou bien rester séparées.

2.4.1 Le test

Supposons que les données à classifier,
$x = (x_1, x_2, \ldots, x_{n_1}, x_{n_1+1}, x_{n_1+2}, \ldots, x_{n_1+n_2})$ soient la réalisation d'un processus de Poisson homogène N dans un domaine convexe D (avec $0 < m(D) < \infty$). On va tester si les données indiquent la présence de deux groupes ou bien d'un seul.

On formule pour cela le test d'hypothèses suivant :

On teste l'hypothèse nulle,

H_0 : Les points $x = (x_1, x_2, \ldots, x_{n_1}, x_{n_1+1}, x_{n_1+2}, \ldots, x_{n_1+n_2})$ sont la réalisation d'un processus de Poisson homogène N sur le domaine D convexe; D est inconnu

contre l'hypothèse alternative,

H_1 : pour une partition $x = (x_1, x_2, \ldots, x_{n_1}, x_{n_1+1}, x_{n_1+2}, \ldots, x_{n_1+n_2})$; on suppose que les points $y = (x_1, x_2, \ldots, x_{n_1})$ sont à l'intérieur d'un domaine convexe D_1 et les points $z = (x_{n_1+1}, x_{n_1+2}, \ldots, x_{n_1+n_2})$ sont dans le domaine convexe D_2; avec $n_1 + n_2 = n$.

On suppose aussi que $D_1 \bigcap D_2 = \emptyset$ et les domaines D_1 et D_2 sont inconnus.

Par les propriétés du processus de Poisson homogène, nous avons que les variables aléatoires y et z sont indépendantes et par la propriété d'uniformité conditionnelle, les points sont distribués de façon uniforme et indépendante dans les domaines D_1 et D_2.

Notre règle de décision va se baser sur l'espace entre les domaines D_1 et D_2.

2.4.2 Le quotient de vraisemblance

Tout comme pour le test des hypervolumes, nous allons considérer le quotient de vraisemblance associé à notre test d'hypothèses.

Par la propriété d'uniformité conditionnelle du processus de Poisson, on peut écrire les fonctions de vraisemblance sous les différentes hypothèses.

Sous H_0, la fonction de vraisemblance s'écrit :

$$\mathcal{L}_D(x) = \frac{1}{(m(D))^n} \prod_{i=1}^{n} I_D(x_i)$$

Sous H_1, la fonction de vraisemblance prend la forme de :

$$\begin{aligned}
\mathcal{L}_D(x) &= \prod_{i=1}^{n_1} \left(\frac{I_{D_1}(x_i)}{m(D_1)+m(D_2)} \right) \times \prod_{i=1}^{n_2} \left(\frac{I_{D_2}(x_i)}{m(D_1)+m(D_2)} \right) \\
&= \frac{1}{[m(D_1)+m(D_2)]^{n_1+n_2}} \times \prod_{i=1}^{n_1} I_{D_1}(x_i) \times \prod_{i=1}^{n_2} I_{D_2}(x_i) \\
&= \frac{1}{[m(D_1)+m(D_2)]^{n}} \times I_{D_1}(H(y)) \times I_{D_2}(H(z))
\end{aligned}$$

Avec $y \equiv (x_1, x_2, \ldots, x_{n_1})$ et $z \equiv (x_{n_1+1}, x_{n_1+2}, \cdots, x_{n_1+n_2})$

On peut donc écrire le quotient de vraisemblance, en se rappelant que les estimateurs du maximum de vraisemblance des différents domaines convexes sont leur enveloppe convexe:

$$\begin{aligned}
Q(x) &= \frac{\max\limits_{H_0} \mathcal{L}_{H_0}(x)}{\max\limits_{H_1} \mathcal{L}_{H_1}(x)} \\
&= \frac{(m_1+m_2)^{n_1+n_2}}{m_D^n} \\
&= \left(\frac{m_1+m_2}{m_D} \right)^n \\
&= \left(1 - \frac{M_1}{m_D} \right)^n
\end{aligned}$$

Où on a utilisé les notations suivantes:

$$\begin{aligned} m_1 &= m(H(y)) \equiv m(H(x_1,x_2,\ldots,x_{n_1})); \\ m_2 &= m(H(z)) \equiv m(H(x_{n_1+1},x_{n_1+2},\ldots,x_{n_1+n_2})); \\ m_D &= m(H(y,z)) \equiv m(H(x_1,x_2,\ldots,x_{n_1},x_{n_1+1},x_{n_1+2},\ldots,x_{n_1+n_2})); \\ M_1 &= m_D - m_1 - m_2. \end{aligned}$$

M_1 représente l'espace entre le domaine D_1 et D_2.

2.4.3 La région critique

On peut écrire la région critique du test de la façon suivante:

$$\left(1 - \frac{M_1}{m_D}\right)^n \leq c$$
$$\Leftrightarrow$$
$$n \log(1 - \frac{M_1}{m_D}) \leq c'$$
$$\Leftrightarrow$$
$$1 - \frac{M_1}{m_D} \leq c''$$
$$\Leftrightarrow$$
$$\frac{M_1}{m_D} \geq c'''$$

Finalement, la région critique est donnée par:

$$\{x \mid \frac{M_1}{m_D} \geq c\}$$

2.4.4 Distribution de la statistique du test

Il nous faut maintenant trouver la distribution de la statistique $\frac{M_1}{m_D}$ du test.

Pour cela, on utilise le résultat suivant (Kubushishi [7]):

$$n\frac{M_1}{m(D)} - \log n - (d-1)\log\log n - \log\kappa = U \text{ quand } n \to \infty$$

Où la variable aléatoire U est telle que:

$$P(U \leq u) = \exp(-\exp(-u))$$

Distribution de la statistique

Pour notre statistique $\frac{M_1}{m_D}$, on aura donc le résultat suivant :

$$\lim_{n\to\infty} P_{H_0}\left(\frac{nM_1}{m_D} - \log n - (d-1)\log\log n - \log\kappa \geq t_\alpha\right)$$

$$= 1 - \lim_{n\to\infty} P_{H_0}\left(\tfrac{nM_1}{m_D} - \log n - (d-1)\log\log n - \log\kappa \leq t_\alpha\right)$$

$$= 1 - \exp[-\exp(-t_\alpha)]$$

On calcule t_α grâce à l'équation :

$$P_{H_0}(T \geq t_\alpha) = \alpha$$

avec $T \equiv n\frac{M_1}{m_D} - \log n - (d-1)\log\log n - \log\kappa$.

$$t_\alpha = -\log[-\log(1-\alpha)]$$

Ainsi, dans notre test, on rejettera l'hypothèse nulle si l'espace entre les classes est grand.

En pratique, on procédera comme ceci, si on considère une partition des données en k classes, pour chaque paire de classe C_i et C_j on calculera la statistique T, notée alors T_{ij}. On récoltera toutes les valeurs de T_{ij}, $i = 1,\ldots,k$ et $j = i+1,\ldots,k$ dans une matrice. Si le minimum de ces valeurs T_{ij} est plus petit que t_α alors on concluera que l'espace entre les classes C_i et C_j pour lesquelles le minimum est atteint n'est pas significatif et on déduira qu'il y a au plus $k-1$ classes naturelles dans les données. En fait on ne rejettera pas l'hypothèse nulle. Pour déterminer le nombre de classes optimal, on réappliquera cette procédure pour la partition en $k-1$ classes, $k-2$ classes, ... Jusqu'à ce qu'on rejette l'hypothèse nulle c'est-à-dire jusqu'à ce que la valeur du minimum des T_{ij} calculés, soit plus grande que t_α.

Calcul de la constante κ

On détermine la constante κ en calculant :

$$\kappa = \frac{1}{d!}\left(\frac{\sqrt{\pi}\,\Gamma(\frac{d}{2}+1)}{\Gamma(\frac{d+1}{2})}\right)^{d-1}$$

Pour de petites dimensions : $\kappa = 1$ pour $d = 1$ et $d = 2$, $\kappa = \frac{3\pi^2}{32}$ pour $d = 3$ et $\kappa = \frac{64}{81}$ pour $d = 4$.

Pour calculer, la constante κ en pratique, il faut se rappeler quelques propriétes de la fonction Gamma.

Si n est un entier, $\Gamma(n) = (n-1)!$ et $\Gamma(n/2) = \frac{(n-2)!!\sqrt{\pi}}{2^{\frac{(n-1)}{2}}}$.

$n!!$ est une double factorielle, c'est-à-dire qu'on aura:

$n!! = n * (n-2) * \cdots * 5 * 3 * 1$ si n est impair
$n!! = n * (n-2) * \cdots * 6 * 4 * 2$ si n est pair.

On a aussi que $\Gamma(1+z) = z\Gamma(z)$.

On calculera donc la constante κ de la façon suivante :

Si d est pair,

$$\Gamma(\tfrac{d}{2}+1) = \tfrac{d}{2}!$$

$$\Gamma(\tfrac{d+1}{2}) = \frac{(d-1)!!\sqrt{\pi}}{2^{\frac{d}{2}}}$$

Si d est impair,

$$\Gamma(\tfrac{d}{2}+1) = \tfrac{d}{2}\Gamma(\tfrac{d}{2})$$

$$= \tfrac{d}{2}\frac{(d-2)!!\sqrt{\pi}}{2^{\frac{(d-1)}{2}}}$$

$$\Gamma(\tfrac{d+1}{2}) = (\tfrac{d+1}{2}-1)!$$

Chapitre 3

Comparaison des résultats pour des données classiques

3.1 Introduction

Nous allons analyser dans ce chapitre, les résultats obtenus après l'application du test des hypervolumes et du gap test à des données classiques. Les partitions de départ sont obtenues en appliquant aux données les méthodes de classification du lien simple, du lien complet, du centroïde et de Ward. Le niveau de signification α pour le gap test a été fixé à 0.05. Pour certains cas, nous avons essayé d'ajuster la valeur de α afin d'obtenir de meilleurs résultats. Pour plusieurs jeux de données nous allons expliquer en détails les résultats obtenus. Pour les autres exemples de données nous serons plus brefs. Avant d'effectuer cette analyse, nous allons présenter brièvement le programme du gap test que nous avons implémenté et le programme du test des hypervolumes. Ces programmes sont écrits en Fortran 77.

3.2 Présentation des programmes

Les deux programmes utilisent la même procédure qui permet de calculer l'enveloppe convexe d'un ensemble de points ainsi que son hypervolume dans des espaces multidimensionnels.

3.2.1 Programme du gap test

Le programme travaille sur une partition des données en un certain nombre de classes suivant une des méthodes hiérarchiques que nous avons

présentées. Cette partition est obtenue par le logiciel SAS [10].

Le programme calcule l'enveloppe convexe de chaque classe et son hypervolume. Il considère ensuite chaque paire de classes possibles et calcule l'envoloppe convexe et l'hypervolume de l'ensemble des points formés par la réunion des deux classes de chaque paire. Ce qui permet ensuite de calculer les valeurs des T_{ij}. On cherche ensuite le minimum de ces valeurs et on le compare avec la valeur de t_α du test. Si le minimum est plus petit que t_α on rejette la partition en k classes considérée.

De plus amples renseignements au sujet de ce programme sont disponibles dans le code (voir syllabus annexe).

3.2.2 Programme du test des hypervolumes

Pour ce programme on utilise également des partitions des données en classes fournies par SAS. Le programme calcule pour chaque classe de la partition son enveloppe convexe et son hypervolume. Il additionne ensuite les k hypervolumes. Pour obtenir la valeur du quotient de vraisemblance, on applique cette procédure pour une partition en k classes et une partition en $k-1$ classes. On effectue ensuite le quotient des deux valeurs obtenues.

D'avantage de renseignements sont également disponibles dans le code du programme (voir syllabus annexe).

3.3 Présentation des résultats

3.3.1 Deux groupes, de tailles inégales en dimension 2

Le premier jeu de données que nous considérons présente deux groupes, un gros et un petit, on peut les représenter dans le graphe suivant (Fig. 3.1). Sur le graphe on distingue bien les deux groupes.

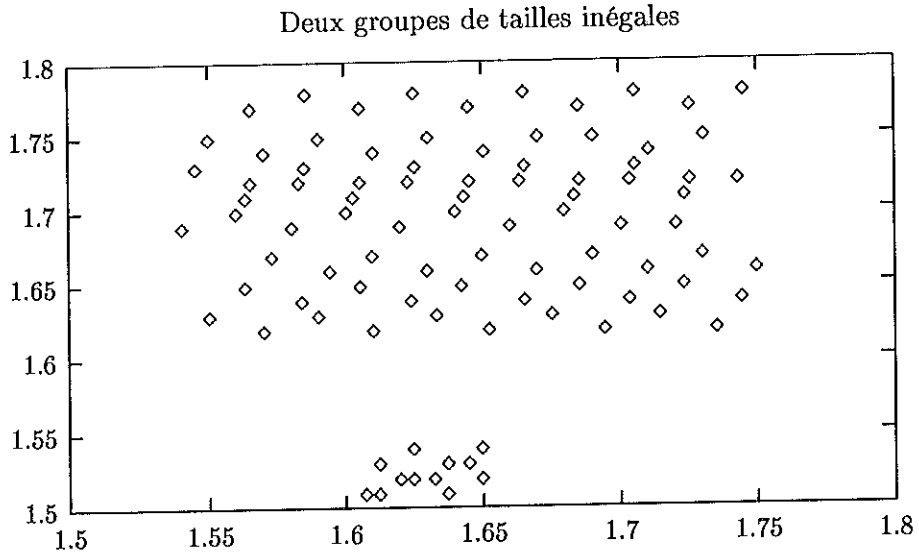


FIG. 3.1 –

Après l'application du gap test et du test des hypervolumes, on obtient les résultats suivants (Tableau 3.1).

La première colonne du tableau nous renseigne sur la méthode de classification utilisée. La deuxième colonne nous dit combien de classes possède la partition considérée. La troisième colonne contient les résultats du gap test, qu'il faut interpréter comme ceci: si l'on considère une partition en 5 classes et que le résultat du gap test correspondant à cette partition est 4, cela signifie que le gap test rejette la partition en 5 classes et nous invite à considérer la partition en 4 classes afin de réappliquer le gap test.

La dernière colonne du tableau contient la valeur du quotient considéré dans le test des hypervolumes. Pour la ligne du tableau qui correspond par exemple, à une partition en 5 classes, la valeur présente dans la quatrième colonne correspondra à la valeur du quotient du test des hypervolumes d'une partition en 5 classes contre une partition en 4 classes.

L'analyse de ces résultats pour notre jeu de données nous apprend que le gap test permet de retrouver le nombre de classes naturelles (i.e. 2). En effet, pour toutes les méthodes de classification il rejette les partitions en 5, 4 et 3 classes pour accepter la partition en 2 classes. De plus la partition en deux classes donnée par toutes les méthodes de classification est la par-

tition naturelle des données en deux classes. En ce qui concerne le test des hypervolumes, on doit rejeter la partition en k classes quand le quotient est proche de 1. On remarque ici que le quotient est proche, voire égal à 1 pour la partition en 3 classes. On conclut donc que selon le test des hypervolumes, le nombre de classes naturelles est 2, ce qui est bien le cas. Pour ce jeu de données, les résultats donnés par les deux méthodes de détermination du nombre de classes sont satisfaisants puisqu'elles retrouvent toutes deux le nombre de classes naturelles présentes dans les données.

Remarquons que pour certaines partitions, la valeur du quotient du test des hypervolumes est égale à 1. Cela s'explique par le fait que les partitions (en 5 classes de la méthode du lien simple et en 3 classes de la méthode du lien complet et du centroïde) ne vérifient pas l'hypothèse que les domaines convexes qu'on considère sont disjoints.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test	Résultats du test des hyper-volumes
Lien simple	7		1
Lien simple	6		0.97
Lien simple	5	4	1
Lien simple	4	3	0.94
Lien simple	3	2	0.99
Lien simple	2	2	0.75
Lien simple	1		
Lien complet	7		0.94
Lien complet	6		0.92
Lien complet	5	4	0.93
Lien complet	4	3	0.84
Lien complet	3	2	1
Lien complet	2	2	0.75
Lien complet	1		
Ward	7		0.94
Ward	6		0.94
Ward	5	4	0.88
Ward	4	3	0.91
Ward	3	2	0.96
Ward	2	2	0.75
Ward	1		
Centroïde	7		0.93
Centroïde	6		0.92
Centroïde	5	4	0.82
Centroïde	4	3	0.85
Centroïde	3	2	1
Centroïde	2	2	0.75
Centroïde	1		

TAB. 3.1 – *Résultats obtenus - Deux groupes de tailles inégales*

3.3.2 Deux groupes parallèles en dimension 2

Encore une fois, nous pouvons représenter les données sur un graphe (Fig. 3.2) et voir qu'elles présentent bien deux groupes.

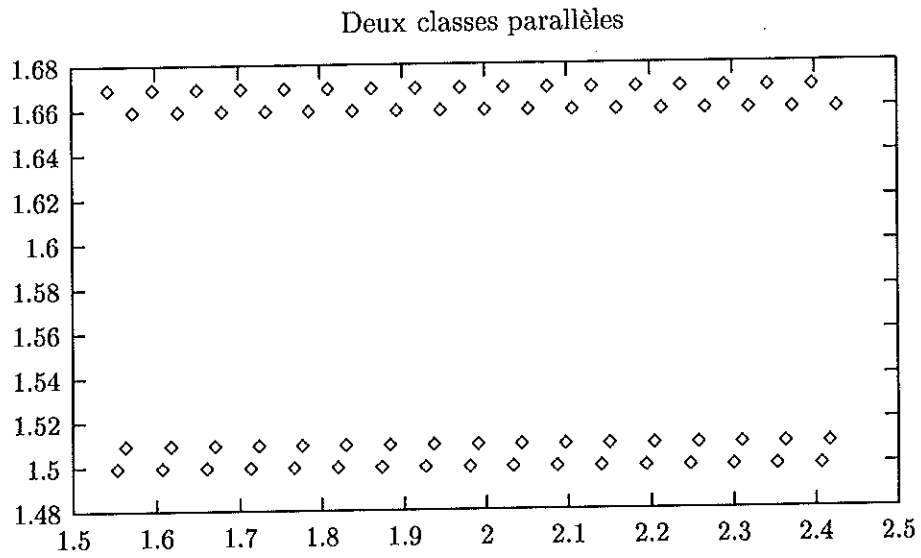


FIG. 3.2 –

Les résultats obtenus après l'application du gap test et du test des hypervolumes au jeu de données sont les suivants (Tableau 3.2).

L'analyse des résultats montre que le test des hypervolumes retrouve la partition en 2 classes uniquement pour la méthode du lien simple. Pour les autres méthodes, on est amené à considérer une classe unique. Ce résultat est tout à fait normal car seule la méthode du lien simple retrouve les classes parallèles. Les autres méthodes ont tendance à former des classes hypersphériques. Les différentes classes des partitions données par ces méthodes seront formées à la fois de points issus du premier groupe et du deuxième groupe tandis que la méthode du lien simple nous donne des partitions de classes qui ne mélangent pas les points des deux groupes parallèles. Les partitions en deux classes obtenues pour les différentes méthodes de classification sont reprises dans les graphes suivants (Fig. 3.3 et Fig. 3.4).

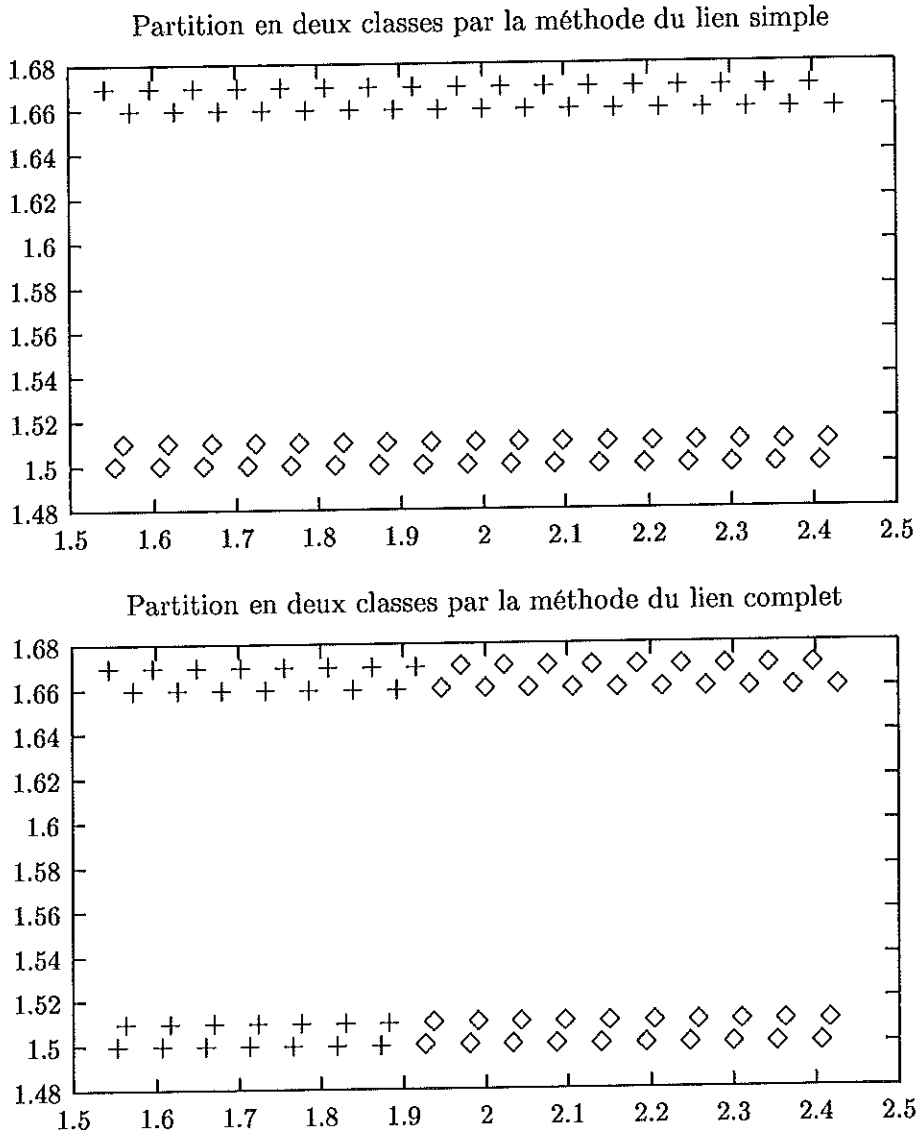


FIG. 3.3 – *Deux classes parallèles*

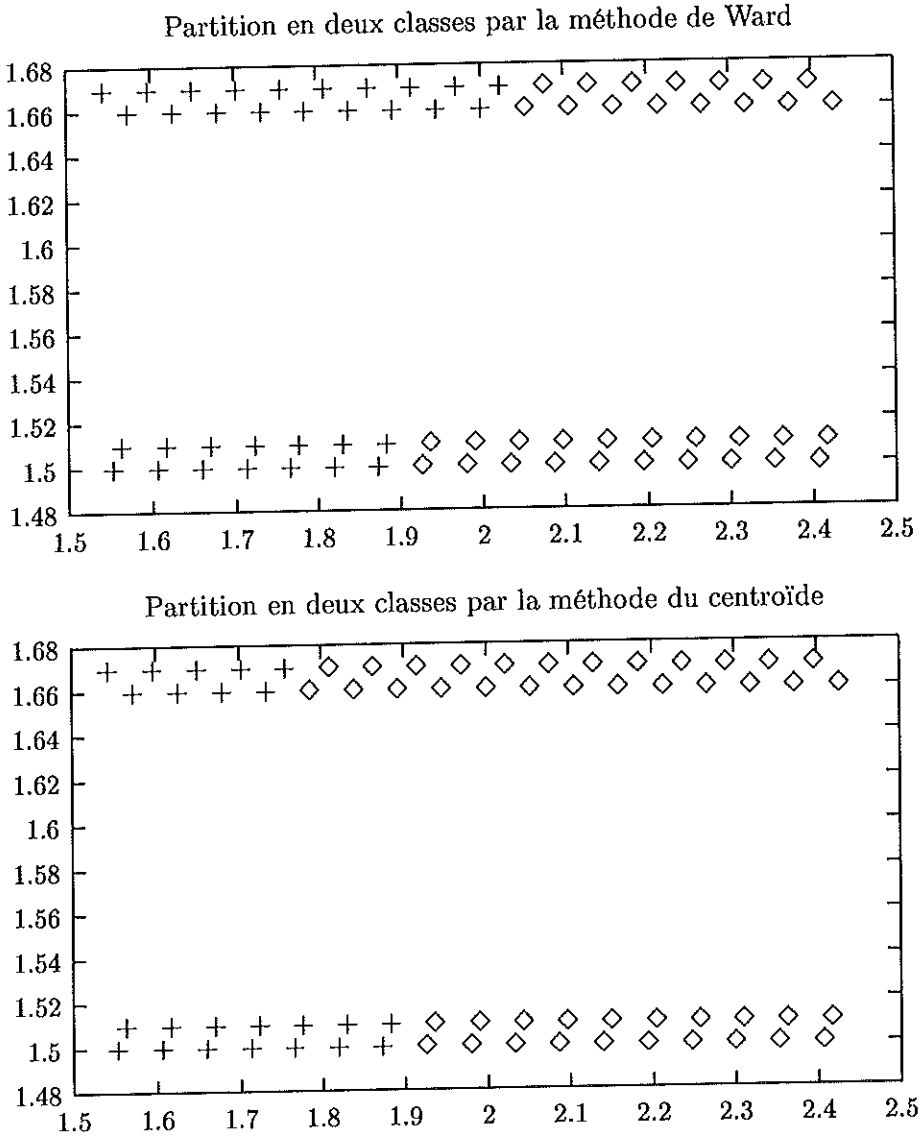


FIG. 3.4 – *Deux classes parallèles*

Les résultats pour le gap test sont identiques. Il retrouve la partition en 2 classes pour la méthode du lien simple et rejette les partitions en 5, 4, 3 et 2 classes pour les autres méthodes de classification.

Pour cet exemple, on conclut que les résultats donnés par les deux tests sont bons car pour la méthode du lien simple ils donnent le nombre de classes naturelles. Pour les autres méthodes, les résultats sont faussés car les partitions données ne reflètent pas la structure parallèle des classes. C'est aussi pourquoi nous n'avons pas essayé d'ajuster la valeur du α du gap test afin d'obtenir de meilleurs résultats pour ces méthodes.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test	Résultats du test des hyper-volumes
Lien simple	7		0.96
Lien simple	6		0.96
Lien simple	5	4	0.96
Lien simple	4	3	0.96
Lien simple	3	2	0.96
Lien simple	2	2	0.11
Lien simple	1		
Lien complet	7		0.88
Lien complet	6		0.81
Lien complet	5	4	0.94
Lien complet	4	3	0.94
Lien complet	3	2	0.95
Lien complet	2	1	0.95
Lien complet	1		
Ward	7		0.57
Ward	6		0.69
Ward	5	4	0.76
Ward	4	3	0.94
Ward	3	2	0.95
Ward	2	1	0.94
Ward	1		
Centroïde	7		0.63
Centroïde	6		0.69
Centroïde	5	4	0.84
Centroïde	4	3	0.84
Centroïde	3	2	0.95
Centroïde	2	1	0.96
Centroïde	1		

TAB. 3.2 – *Résultats obtenus - Deux groupes parallèles*

3.3.3 Trois classes parallèles en dimension 3

Ici encore, la partition naturelle des données en trois classes ne se trouvera pas dans la hiérarchie de partitions donnée par les méthodes du lien complet, de Ward et du centroïde étant donné leur tendance à former des groupes hypersphériques. Par contre on la retrouvera dans la partition en trois classes donnée par la méthode du lien simple. C'est pourquoi, les deux tests retrouvent la partition en 3 classes uniquement pour cette méthode.

Pour les autres méthodes, les résultats des tests sont moins bons. Le gap test accepte la partition en 4 et 3 classes pour ensuite refuser la partition en 2 classes pour la méthode du lien complet. Il accepte les partitions en 4, 3 et 2 classes pour la méthode de Ward et il accepte la partition en 2 classes pour la méthode du centroïde.

Pour le test des hypervolumes, on conclut en l'absence de structure pour les méthodes du lien complet et de Ward. Il trouve 2 classes pour la méthode du centroïde.

3.3.4 Trois classes parallèles en dimension 2

On considère aussi trois classes parallèles mais en dimension 2. Les données sont représentées sur le graphe suivant (Fig. 3.5).

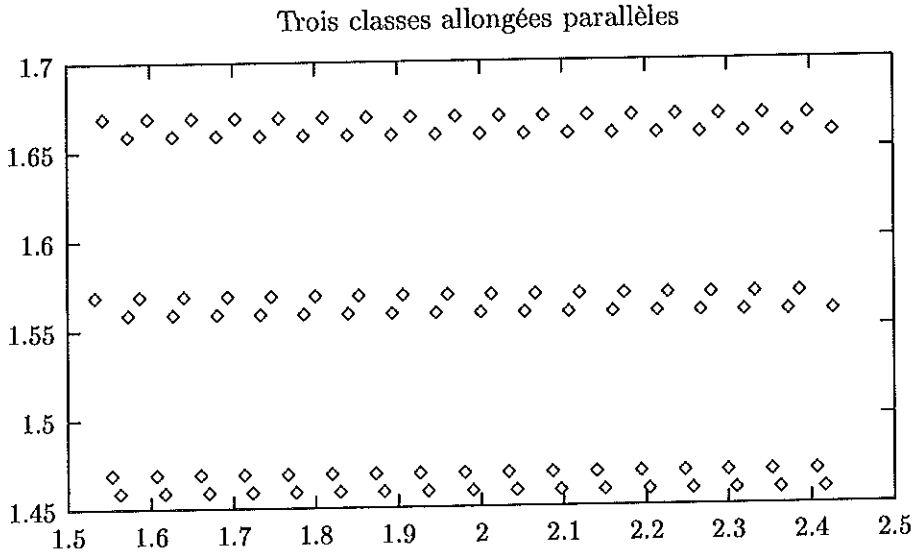


FIG. 3.5 –

Pour les deux tests les conclusions sont identiques. Ils retrouvent tous deux la partition en 3 classes uniquement pour la méthode du lien simple. Pour les autres méthodes, le gap test conclut qu'il n'y a pas de structure dans les données et le test des hypervolumes nous mène à la même conclusion. Encore une fois et pour la même raison que dans le cas précédent, on pouvait s'attendre à ces résultats. On peut donc en conclure que ceux-ci sont satisfaisants. Les partitions en 3 classes pour les différentes méthodes de classification sont présentées dans les graphes suivants (Fig. 3.6 et Fig. 3.7).

De même que pour le cas de deux classes parallèles, dans ce cas-ci et le précédent, nous n'avons pas jugé utile d'essayer d'ajuster la valeur du α du gap test afin d'obtenir de meilleurs résultats pour les méthodes de classification du lien complet, de Ward et du centroïde.

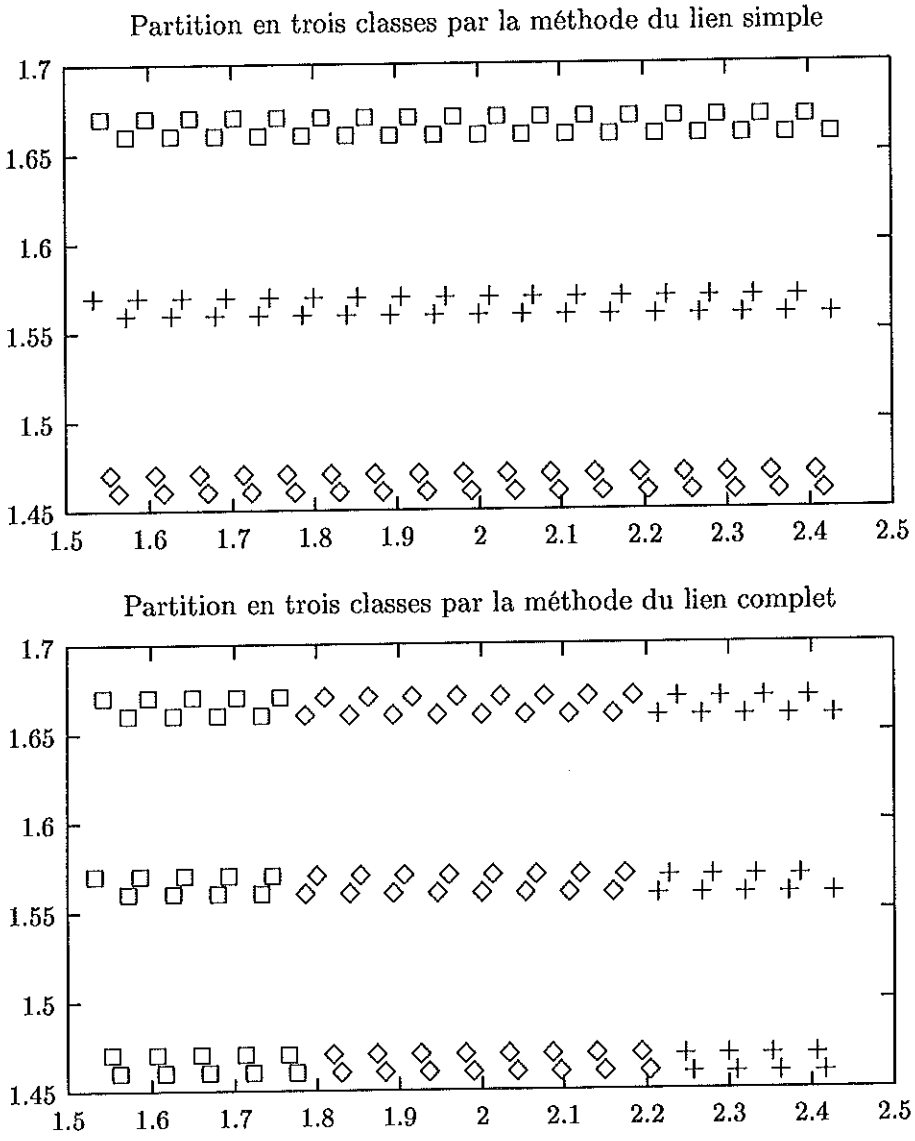


FIG. 3.6 – *Trois classes parallèles*

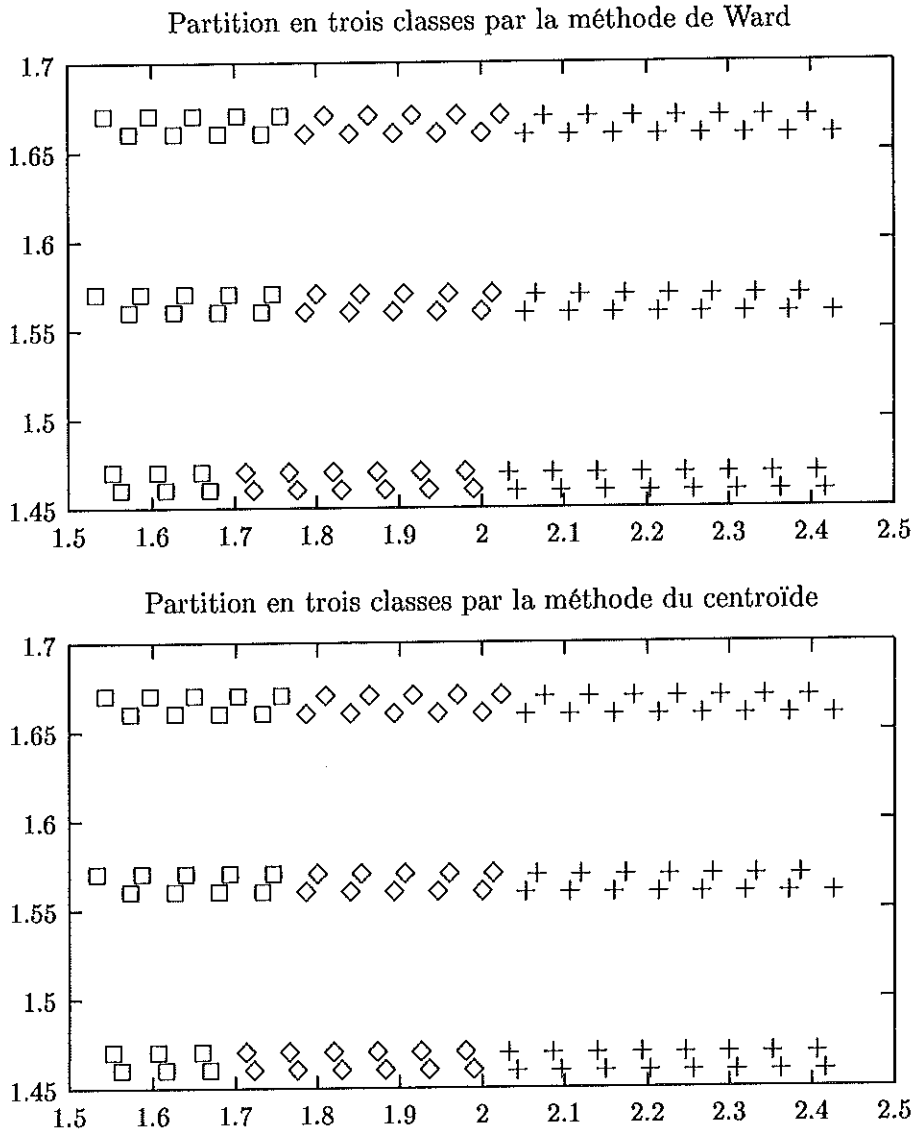


FIG. 3.7 – *Trois classes parallèles*

3.3.5 Deux groupes éloignés en dimension 3

Pour ce jeu de données, nous obtenons les résultats suivants (Tableau 3.3). Le test des hypervolumes retrouve le nombre de classes naturelles pour toutes les méthodes de classification. Le gap test trouve 2 classes pour les partitions obtenues par la méthode du lien simple. Le gap test refuse la partition en 3 classes donnée par la méthode du centroïde; par contre si on l'applique avec une partition en 5 ou 4 classes, il les acceptera. Pour les deux autres méthodes, le gap test refuse la partition en 5 classes mais accepte les partitions en 4 et 3 classes. Pour cet exemple, il est difficile d'expliquer les mauvais résultats du gap test pour ces méthodes.

Le graphe des données en trois dimensions est le suivant (Fig. 3.8). Remarquons que toutes les méthodes de classification donnent comme partition en deux classes la partition naturelle.

Deux groupes éloignés en dimension 3

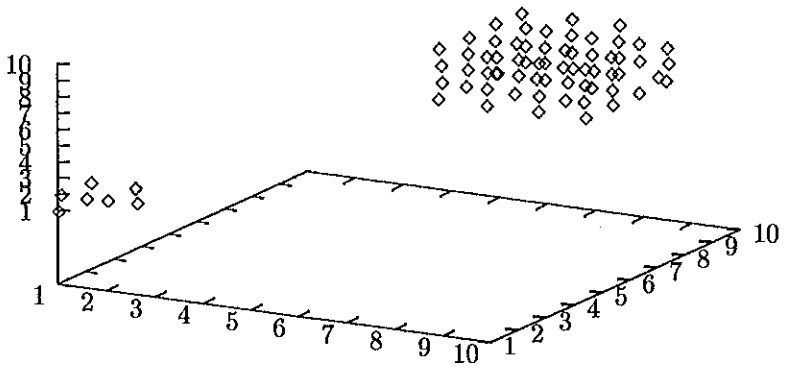


FIG. 3.8 –

Dans le tableau, nous avons aussi indiqué les valeurs de α pour lesquelles le gap test donne les bons résultats: il s'agit de la deuxième colonne du tableau concernant les résultats du gap test. On remarque que ces valeurs doivent être extrêmement petites pour qu'on obtienne un meilleur résultat. L'ajustement du degré de signification n'influence donc pas grandement les résultats du test dans ce cas.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7			0.98
Lien simple	6			0.94
Lien simple	5	4		0.98
Lien simple	4	3		0.98
Lien simple	3	2		0.99
Lien simple	2	2		0.32
Lien simple	1			
Lien complet	7			0.96
Lien complet	6			0.96
Lien complet	5	4		0.92
Lien complet	4	4	0.0045	0.81
Lien complet	3	3	0.00015	0.76
Lien complet	2	2		0.32
Lien complet	1			
Ward	7			0.87
Ward	6			0.81
Ward	5	4		0.88
Ward	4	4	0.01	0.81
Ward	3	3	0.015	0.79
Ward	2	2		0.32
Ward	1			
Centroïde	7			0.63
Centroïde	6			0.69
Centroïde	5	5	0.035	0.87
Centroïde	4	4	0.000047	0.70
Centroïde	3	2		0.87
Centroïde	2	2		0.32
Centroïde	1			

TAB. 3.3 – *Résultats obtenus - Deux groupes éloignés en dimension 3*

3.3.6 Deux groupes proches en dimension 3

Pour ces données, le test des hypervolumes retrouve la partition en 2 classes pour toutes les méthodes de classification. Pour la méthode du gap test, on retrouve la partition en 2 classes pour la méthode du lien simple et la méthode du centroïde. Pour la méthode du lien complet, le gap test accepte les partitions en 5, 4, 3 et 2 classes. Dans le cas de la méthode de Ward, le test accepte la partition en 5 et 4 classes mais refuse la partition en 3 classes au profit de la partition en 2 classes. Encore une fois il est difficile d'expliquer les mauvais résultats du gap test. Le graphe représentant les données est le suivant (Fig. 3.9). Tout comme dans le cas précédent, toutes les méthodes de classification donnent comme partition des données en deux classes la partition naturelle.

Deux groupes proches en dimension 3

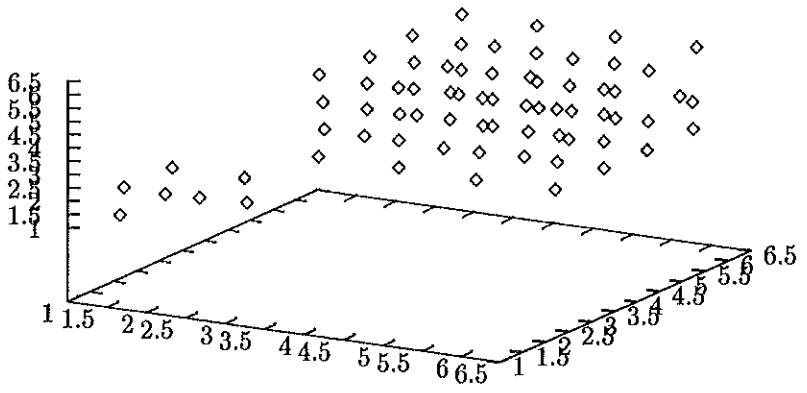


FIG. 3.9 –

Sur le tableau suivant (Tableau 3.4), nous avons recueilli les résultats obtenus ainsi que l'ajustement du α pour obtenir de meilleurs résultats. Les valeurs de α doivent être petites pour obtenir de bons résultats. On remarquera cependant que si on fixe α à 0.01 pour tester si les données présentent trois classes, le gap test rejettera les partitions en trois classes pour toutes les méthodes. On trouvera donc 2 comme nombre de classes naturelles.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7			1
Lien simple	6			0.99
Lien simple	5	4		0.98
Lien simple	4	3		0.99
Lien simple	3	2		0.97
Lien simple	2	2		0.63
Lien simple	1			
Lien complet	7			0.91
Lien complet	6			0.81
Lien complet	5	5	0.0067	0.83
Lien complet	4	4	0.003	0.79
Lien complet	3	3	0.01	0.82
Lien complet	2	2		0.61
Lien complet	1			
Ward	7			0.87
Ward	6			0.82
Ward	5	5	0.0067	0.81
Ward	4	4	0.0028	0.72
Ward	3	2		0.98
Ward	2	2		0.61
Ward	1			
Centroïde	7			0.89
Centroïde	6			0.87
Centroïde	5	4		0.74
Centroïde	4	3		0.78
Centroïde	3	2		0.94
Centroïde	2	2		0.61
Centroïde	1			

TAB. 3.4 – *Résultats obtenus - Deux groupes proches en dimension 3*

3.3.7 Trois groupes bien séparés en dimension 2

Pour ces données nous avons le graphe suivant (Fig. 3.10). Le test des hypervolumes retrouve la partition en 3 classes pour toutes les méthodes de classification.

Le gap test retrouve la partition en 3 classes pour les méthodes du lien simple, du lien complet et du centroïde. Pour la méthode de Ward, le gap test accepte d'abord la partition en 4 classes avant d'accepter les partitions en 3 et 2 classes. Remarquons que la partition en 3 classes pour toutes les méthodes de classification est la partition naturelle des données. Les résultats sont donc relativement bons. Pour obtenir un résultat parfait et donc que le gap test refuse la partition en 4 classes de la méthode Ward, il faut fixer α à 0.028.

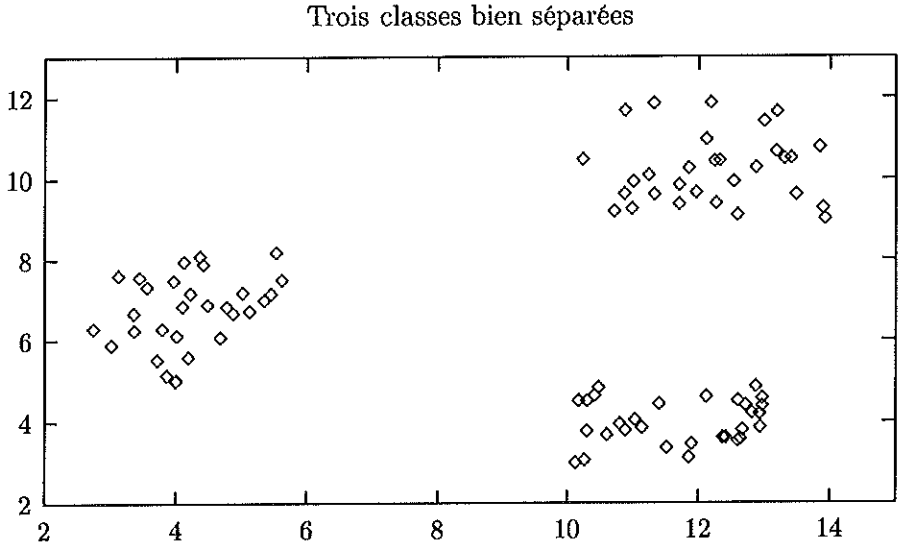


FIG. 3.10 –

3.3.8 Deux classes bien séparées en dimension 2

Pour ces données nous obtenons le graphe suivant (Fig. 3.11). Le test des hypervolumes retrouve la partition en 2 classes pour toutes les méthodes.

Pour la méthode du lien simple et du centroïde, le gap test retrouve la partition en 2 classes mais pour la méthode du lien complet, le test accepte la partition en 5 classes pour ensuite accepter la partition en 2 classes. Pour la méthode de Ward, le gap test accepte la partition en 3 classes avant d'accepter la partition en 2 classes. Ici encore, la partition en deux classes pour toutes les méthodes de classification est la partition naturelle des données en deux classes. Pour obtenir que le gap test trouve 2 comme nombre de classes naturelles pour toutes les méthodes de classification, il faut fixer α à 0.001

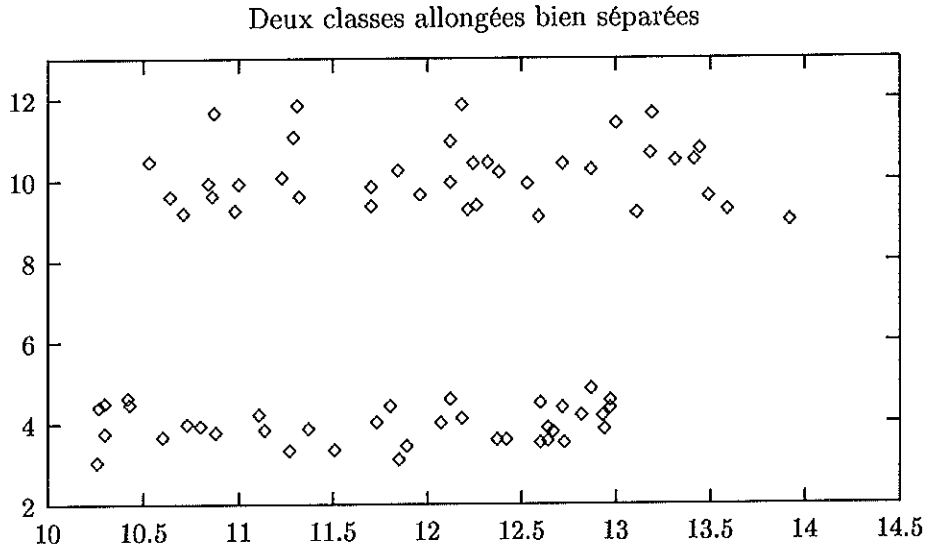


FIG. 3.11 –

3.3.9 Trois classes allongées en dimension 2

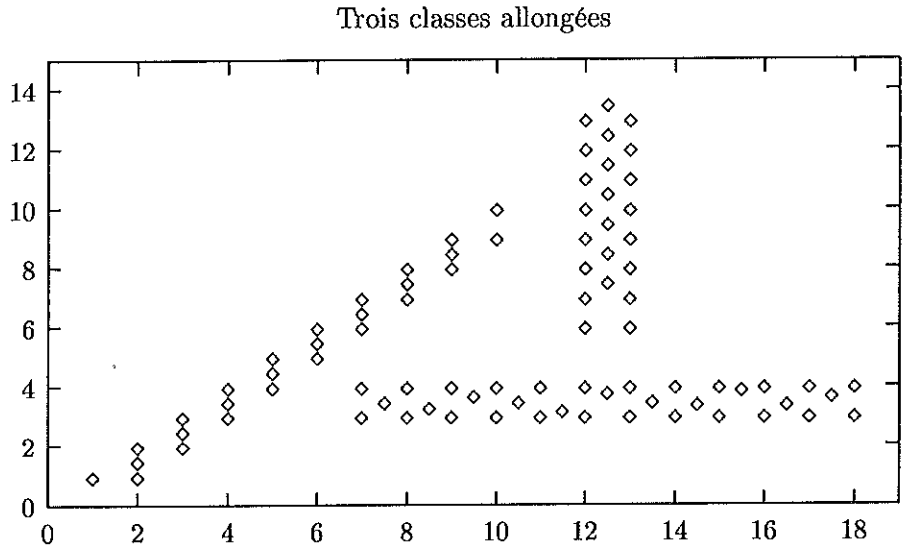


FIG. 3.12 –

On a pu tester deux jeux de données d'effectifs différents qui présentaient ces caractéristiques. Le graphe (Fig. 3.12) présente un de ces jeux de données.

On peut remarquer sur le graphe représentant les données que les classes ne sont pas séparables par des hyperplans et que l'espace entre les classes n'est pas grand. De plus, rappelons que les méthodes du lien complet, de Ward et du centroïde ont tendance à former des groupes hypersphériques. Les classes de cet exemple étant allongées, seule la méthode du lien simple permet de retrouver la partition naturelle en trois classes comme le montrent les graphes suivants (Fig. 3.13 et Fig. 3.14). Cela explique les résultats donnés par le test des hypervolumes. Les résultats du gap test sont plus difficiles à interpréter.

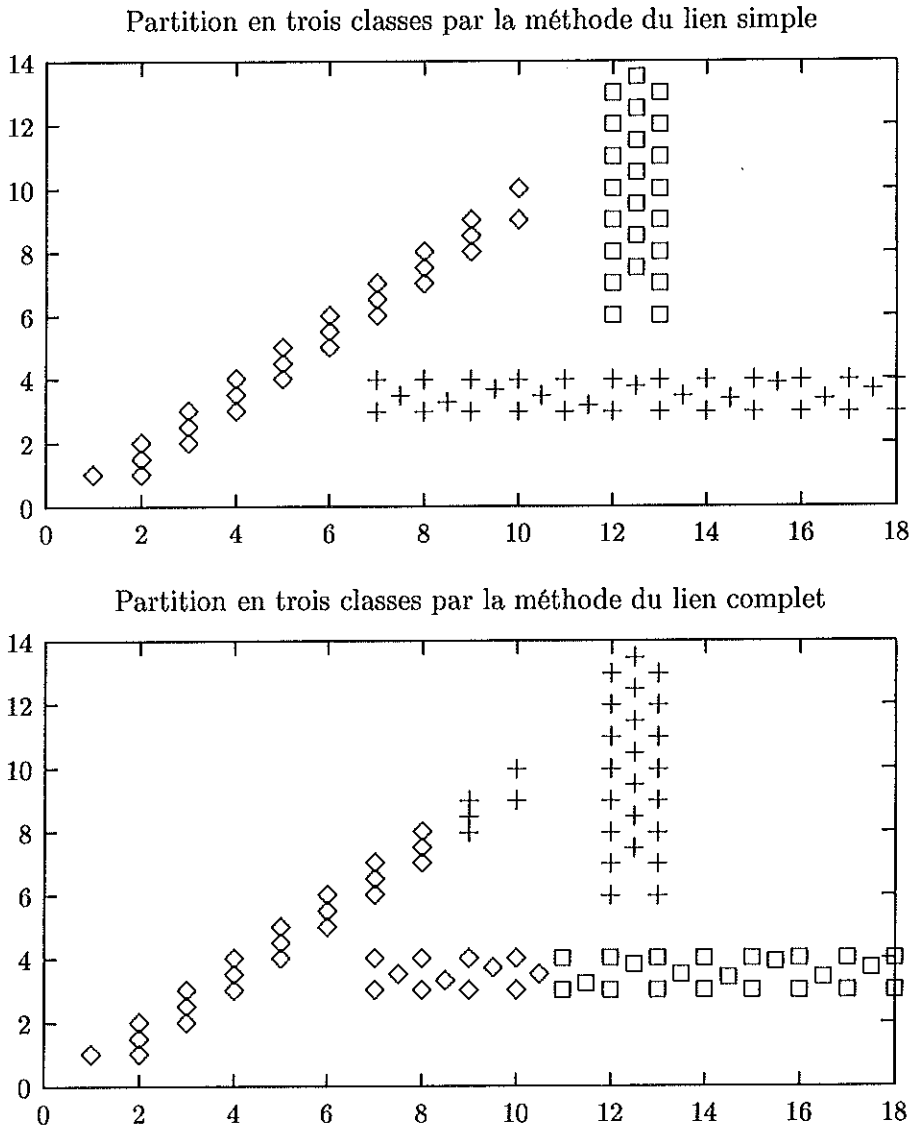


FIG. 3.13 – *Trois classes allongées*

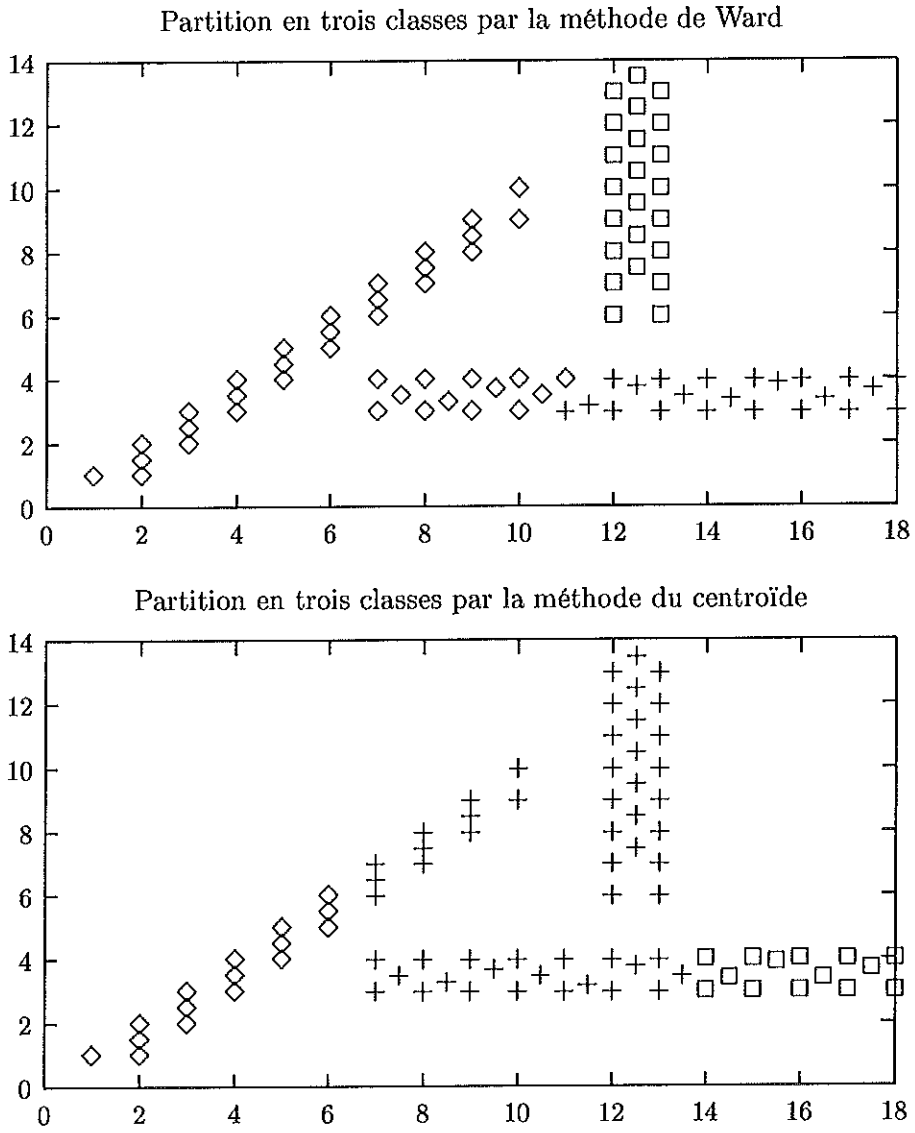


FIG. 3.14 – *Trois classes allongées*

Pour le test des hypervolumes, on retrouve pour les deux jeux de données la partition en 3 classes uniquement pour la méthode du lien simple, pour les autres méthodes on conclut en l'absence de structure.

Dans le cas du gap test, les résultats sont différents pour les deux jeux de données. Pour le premier, le gap test retrouve la partition en 3 classes pour toutes les méthodes de classification. Mais il est important de remarquer que seule la partition en trois classes donnée par la méthode du lien simple est la partition naturelle des données en trois classes. Pour le deuxième jeu de données (qui présente plus de points que le premier), le test retrouve la partition en 3 classes donnée par la méthode du lien simple et par la méthode du centroïde. Cependant, la partition en trois classes donnée par la méthode du centroïde n'est pas la partition naturelle contrairement à la partition donnée par la méthode du lien simple. Pour les deux autres méthodes le gap test accepte aussi la partition en 4 classes.

3.3.10 Données non séparables en dimension 3

Remarquons que pour ce jeu de données, les trois classes naturelles ne sont pas séparables par un hyperplan ce qui explique les mauvais résultats des deux tests. Pour ce jeu de données, le test des hypervolumes conclut en l'absence de structure pour toutes les méthodes de classification. Le gap test mène à cette même conclusion pour la méthode du lien simple. Pour les autres méthodes, il accepte toutes les partitions en 5, 4, 3 et 2 classes. Les données sont présentées dans le graphe à trois dimensions suivant (Fig. 3.15).

Données non séparables en trois dimensions

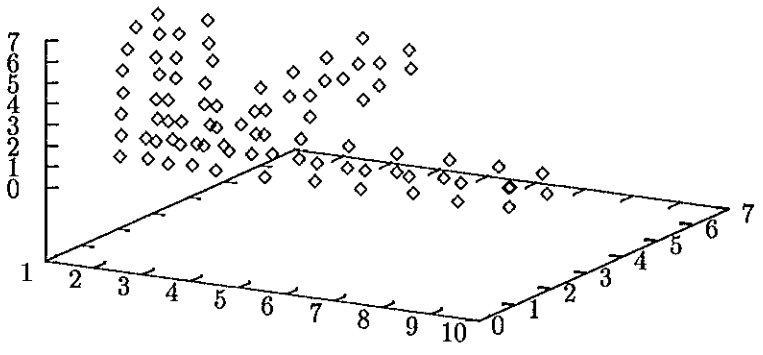


FIG. 3.15 –

Remarquons qu'il faut fixer α à des valeurs extrêmement petites pour obtenir de meilleurs résultats pour le gap test.

3.3.11 Données sans structure en dimension 2

Nous pouvons nous représenter les données dans le graphe suivant (Fig. 3.16).

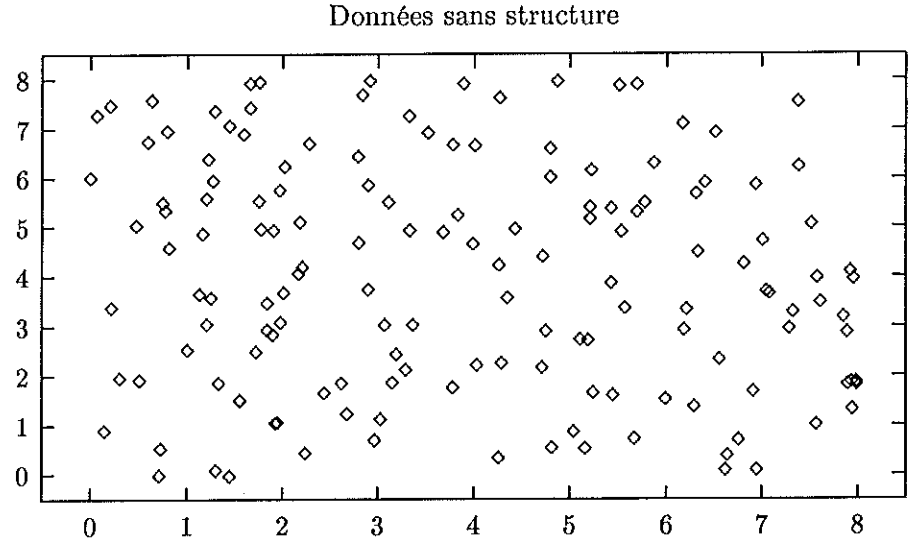


FIG. 3.16 –

Le test des hypervolumes nous permet de conclure qu'il n'y a pas de structure dans les données et ce pour toutes les méthodes de classification.

Le gap test refuse toutes les partitions en 5, 4, 3 et 2 classes uniquement pour la méthode du lien simple. Pour la méthode du lien complet, le test accepte la partition en 3 classes pour ensuite refuser la partition en 2 classes. Le test accepte la partition en 5 classes dans le cas de la méthode de Ward pour ensuite refuser les autres partitions en 4, 3 et 2 classes. Pour la méthode du centroïde, le test accepte la partition en 3 et en 2 classes. Dans ce cas-ci, le gap test nous donne de bons résultats uniquement pour la méthode du lien simple. Il faut fixer la valeur de α à 0.0002 pour obtenir les bons résultats pour les autres méthodes.

3.3.12 Données de Ruspini, quatre classes en dimension 2

Nous avons également appliqué nos deux tests aux données de Ruspini représentées dans le graphe suivant (Fig. 3.17).

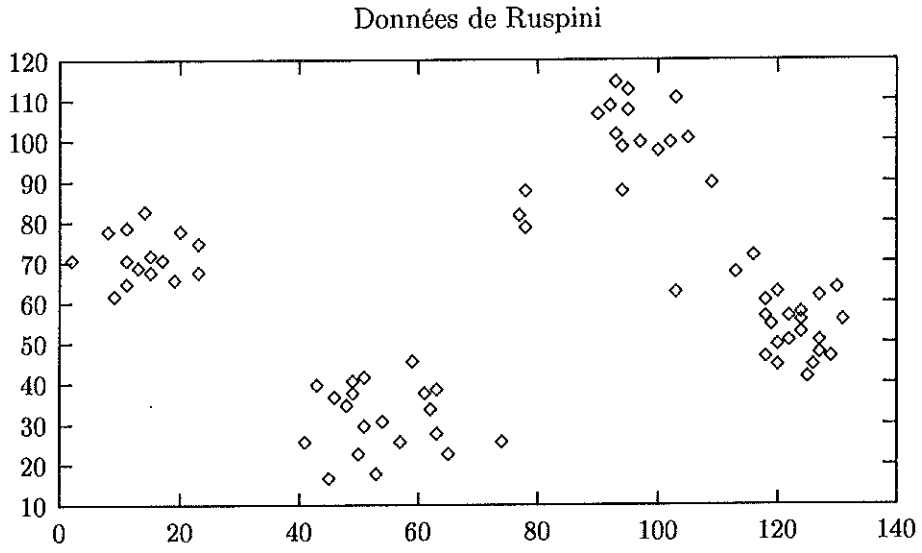


FIG. 3.17 –

Il est difficile de conclure sur le nombre de classes par le test des hypervolumes, cependant il semble retrouver les quatre classes pour toutes les méthodes de classification sauf pour la méthode du lien complet. Les partitions en 4 classes et 5 classes sont les mêmes pour toutes les méthodes de classification sauf pour celle du lien complet. Ce qui explique les résultats du test des hypervolumes. De plus, la partition en 4 classes donnée par ces méthodes (lien simple, Ward et centroïde) est la partition naturelle des données. Les graphes suivants (Fig. 3.18 et Fig. 3.19) montrent les partitions en 4 classes et 5 classes pour les méthodes du lien simple et du lien complet.

Le gap test retrouve la partition en 4 classes uniquement pour la méthode du lien complet or ce n'est pas la partition naturelle en quatre classes. Pour les autres méthodes, il accepte d'abord la partition en 5 classes pour le lien simple et Ward et la partition en 6 classes pour la méthode du centroïde. Il est difficile d'expliquer pourquoi nous obtenons de mauvais résultats. Dans

le tableau suivant (Tableau 3.5), nous avons recueilli nos résultats ainsi que les valeurs de α que nous avons ajustées pour les améliorer. Ces valeurs sont assez petites.

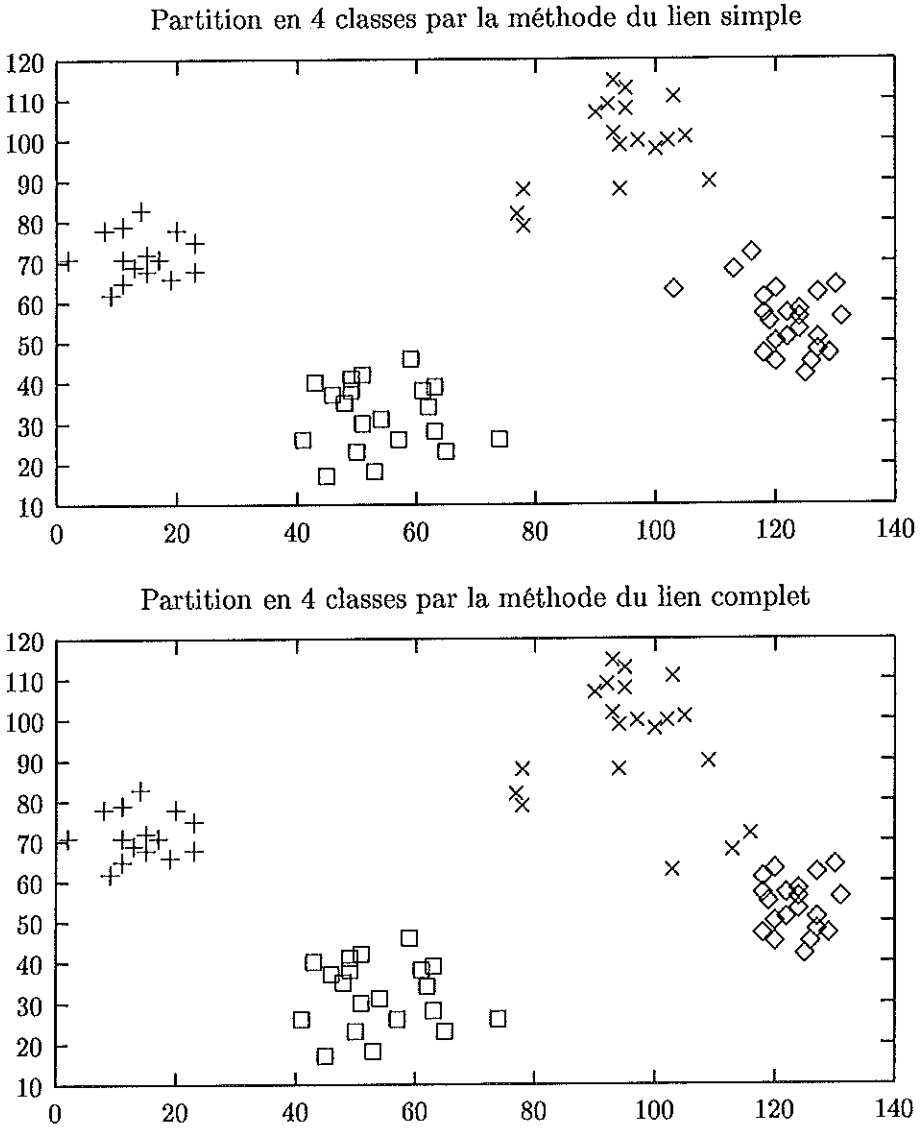


FIG. 3.18 – *Données de Ruspini*

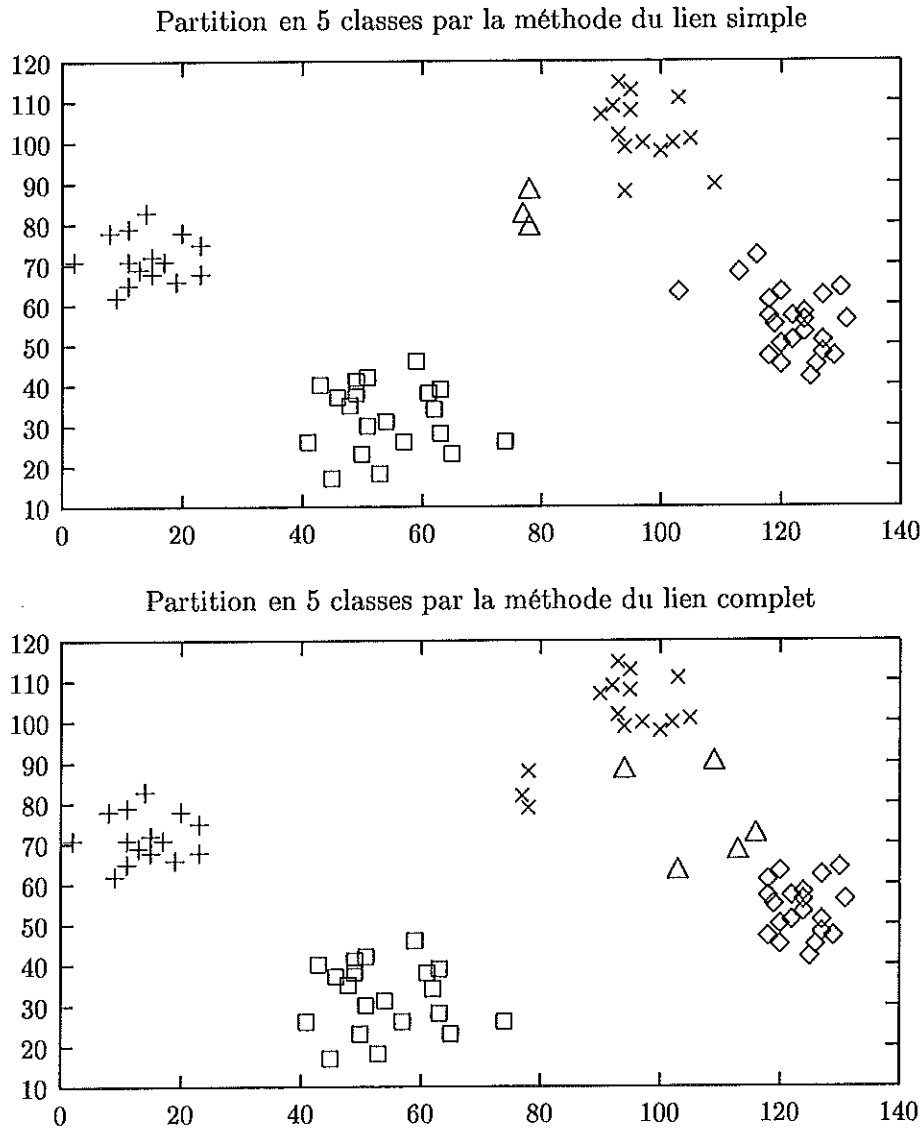


FIG. 3.19 – *Données de Ruspini*

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7	6		0.91
Lien simple	6	5		0.94
Lien simple	5	5	0.015	0.85
Lien simple	4	4	0.0000035	0.69
Lien simple	3	3		0.73
Lien simple	2	2		0.49
Lien simple	1			
Lien complet	7	6		0.90
Lien complet	6	5		0.86
Lien complet	5	4		0.82
Lien complet	4	4		0.80
Lien complet	3	3		0.73
Lien complet	2	2		0.49
Lien complet	1			
Ward	7	6		0.91
Ward	6	5		0.88
Ward	5	5	0.015	0.85
Ward	4	4		0.69
Ward	3	3		0.73
Ward	2	2		0.49
Ward	1			
Centroïde	7	6		0.89
Centroïde	6	6	0.03	0.86
Centroïde	5	5	0.015	0.85
Centroïde	4	4		0.69
Centroïde	3	3		0.73
Centroïde	2	2		0.49
Centroïde	1			

TAB. 3.5 – *Résultats obtenus - Données de Ruspini*

3.3.13 Deux classes non convexes en dimension 2

Les données que nous considérons ici (Fig. 3.20) ne vérifient pas l'hypothèse de convexité des domaines pour les classes naturelles. Les résultats obtenus sont les suivants (Tableau 3.6).

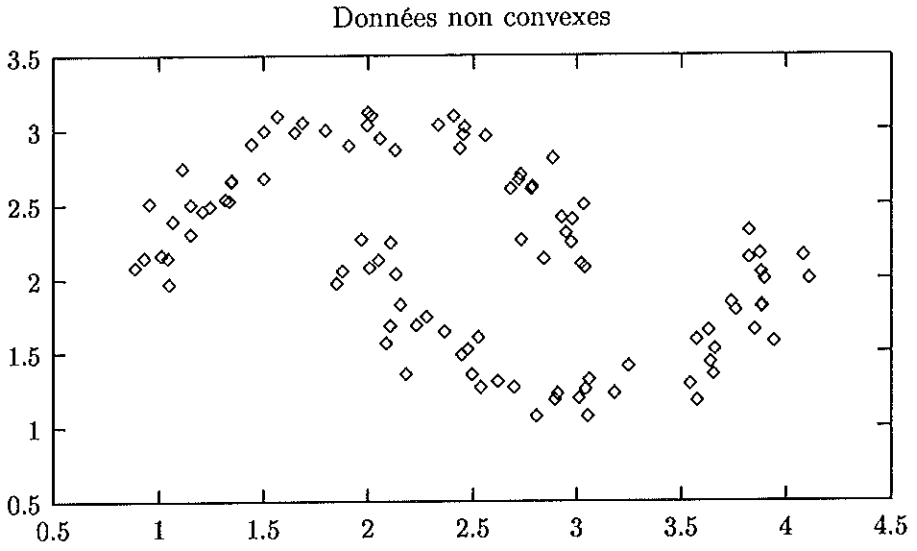


FIG. 3.20 –

Remarquons que seule la méthode du lien simple nous donne la bonne partition en 2 classes, comme le montre les graphes suivants (Fig. 3.21 et Fig. 3.22).

Le test des hypervolumes conclut en l'absence de structure pour toutes les méthodes de classification. Dans le cas du gap test, pour la méthode du lien simple, il rejette la partition en 5 classes et en 2 classes mais accepte les partitions en 4 et 3 classes. Pour les autres méthodes, le test accepte les partitions en 5, 4, 3 et 2 classes. Il est donc difficile de conclure sur le nombre de classes naturelles présentes dans les données. De plus, si on observe les valeurs ajustées de α, on remarque qu'elles sont très petites. Pour ces données, les résultats sont donc très mauvais.

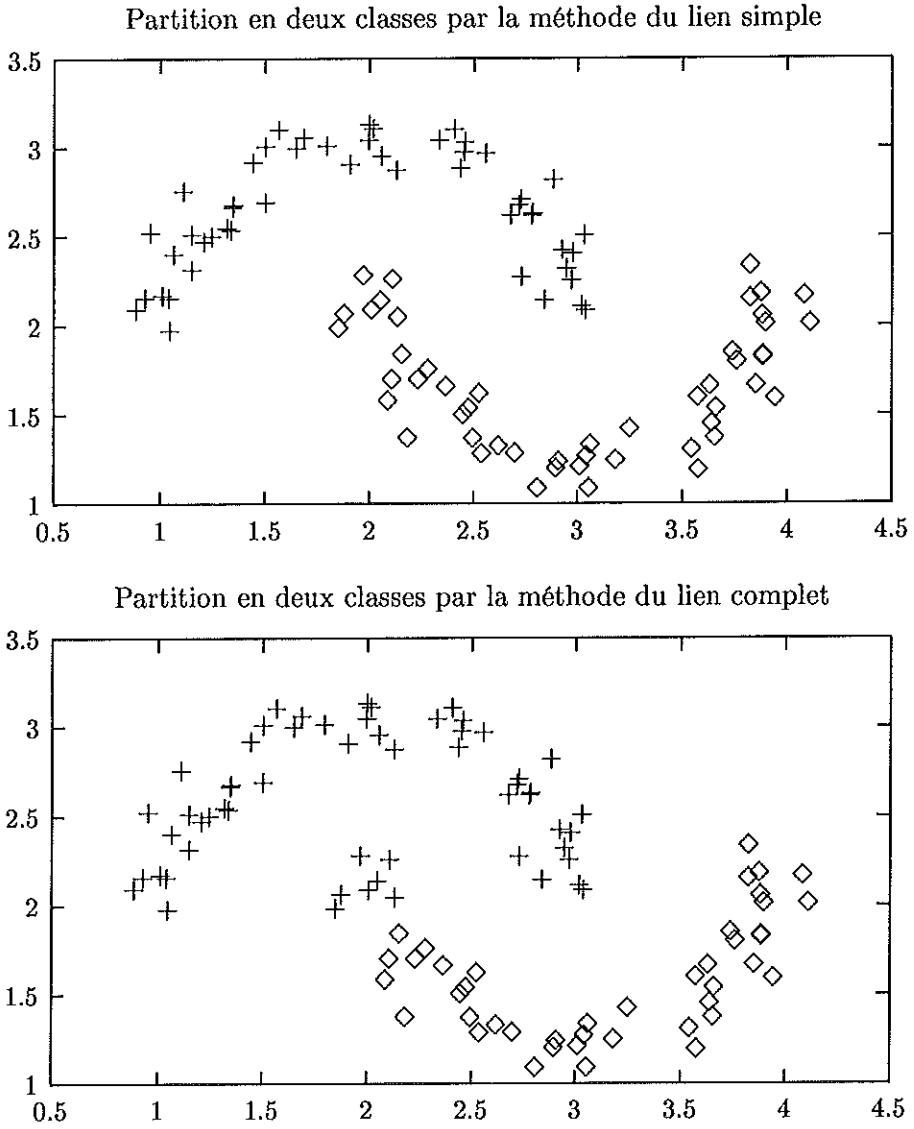


FIG. 3.21 – *Classes non convexes*

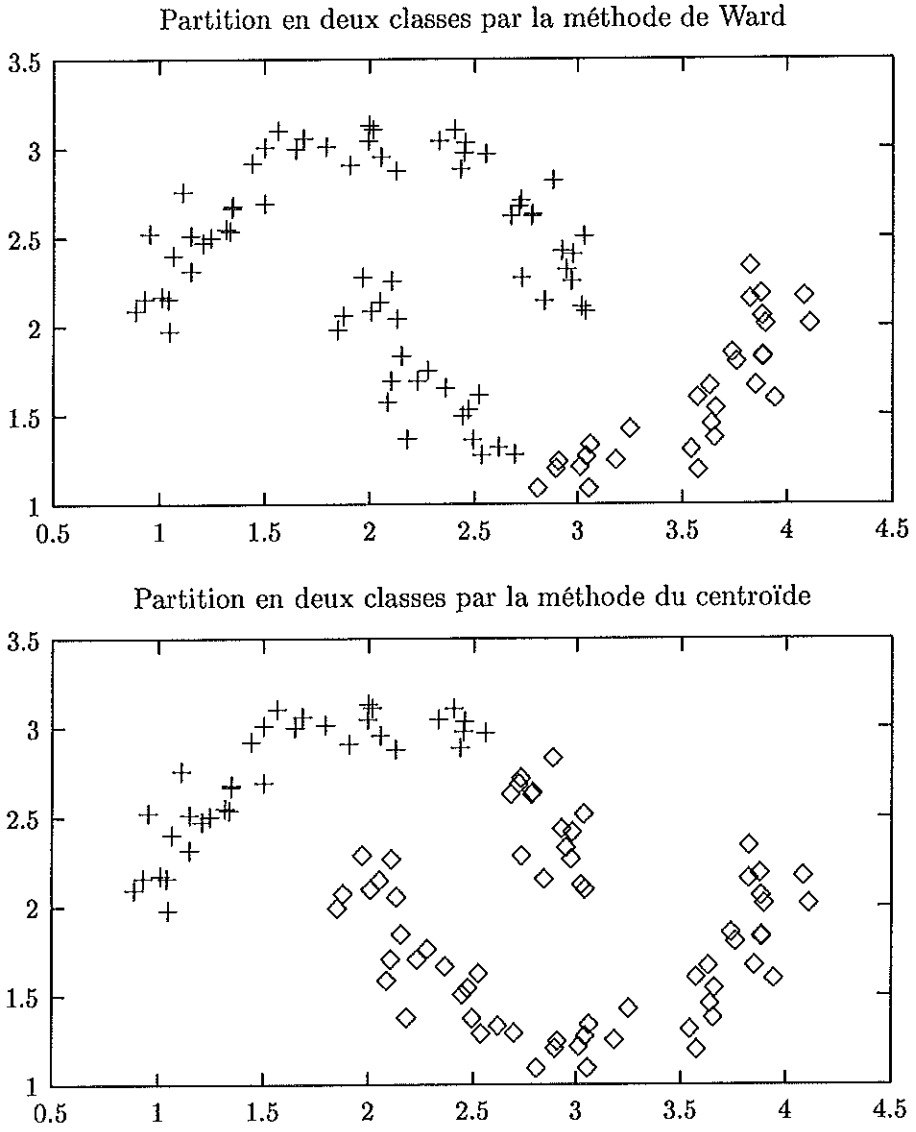


FIG. 3.22 – *Classes non convexes*

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7			0.95
Lien simple	6			0.98
Lien simple	5	4		0.90
Lien simple	4	4	0.00002	0.72
Lien simple	3	3	0.005	0.75
Lien simple	2	1		0.95
Lien simple	1			
Lien complet	7			0.91
Lien complet	6			0.80
Lien complet	5	5	0.01	0.84
Lien complet	4	4	0.0001	0.73
Lien complet	3	3	0.0001	0.86
Lien complet	2	2		0.84
Lien complet	1			
Ward	7			0.89
Ward	6			0.82
Ward	5	5	0.0004	0.85
Ward	4	4	$<<<$	0.74
Ward	3	3	0.000005	0.80
Ward	2	2		0.82
Ward	1			
Centroïde	7			0.89
Centroïde	6			0.79
Centroïde	5	5	0.00002	0.76
Centroïde	4	4	0.00002	0.83
Centroïde	3	3	0.000002	0.79
Centroïde	2	2		0.82
Centroïde	1			

TAB. 3.6 – *Résultats obtenus - Classes non convexes*

3.3.14 Quatre classes non convexes en dimension 3

On considère ici des données présentant des classes qui ne respectent pas l'hypothèse de convexité des domaines pour les classes naturelles mais en dimension 3. Encore une fois, seule la méthode du lien simple nous donne la bonne partition en 4 classes. Les graphes suivants (Fig. 3.23 et Fig. 3.24) montrent les différentes partitions en 4 classes obtenues pour les méthodes de classification utilisées.

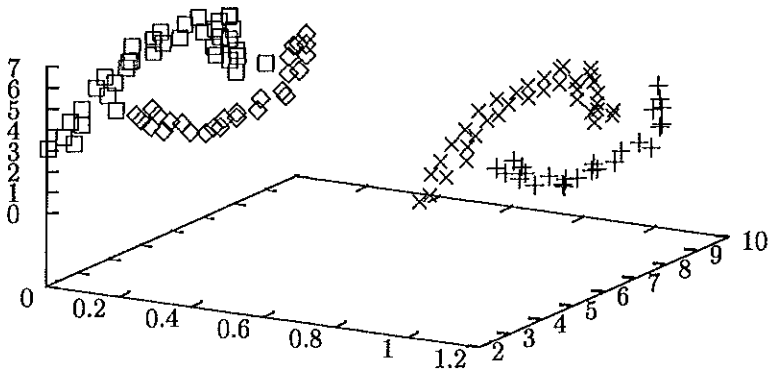
Les résultats obtenus pour les deux tests sont présentés dans le tableau suivant (Tableau 3.7).

Le test des hypervolumes retrouve la partition en 4 classes donnée par la méthode du lien simple. Pour les autres méthodes on conclut en l'absence de structure.

Le gap test refuse la partition en 6 classes pour la méthode du lien simple. Il accepte toutes les autres partitions en 6, 5, 4, 3 et 2 classes pour toutes les autres méthodes, on ne sait donc pas conclure.

Les valeurs ajustées du α sont, comme dans le cas précédent, très petites. Dans l'ensemble les résultats ne sont donc pas satisfaisants.

Partition en 4 classes par la méthode du lien simple



Partition en 4 classes par la méthode du lien complet

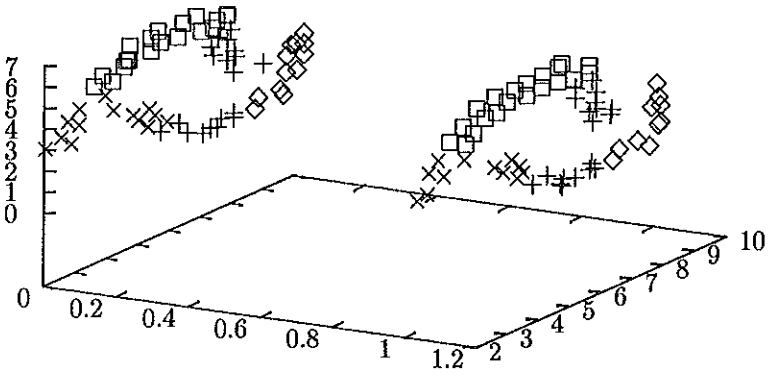
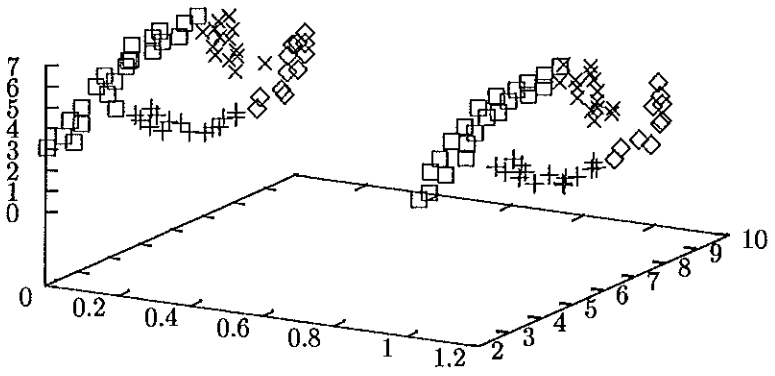


FIG. 3.23 – *Classes non convexes*

Partition en 4 classes par la méthode de Ward



Partition en 4 classes par la méthode du centroïde

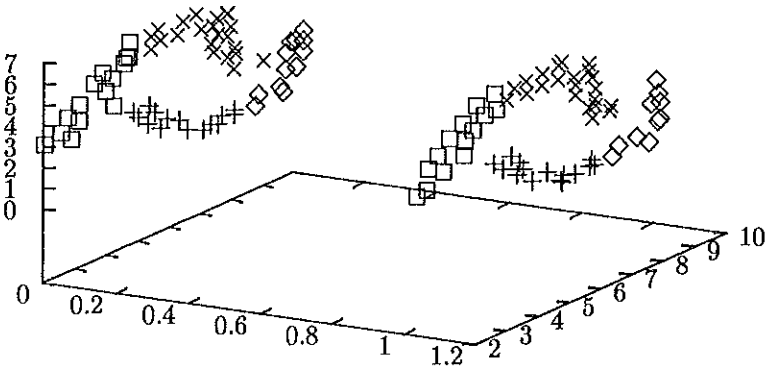


FIG. 3.24 – *Classes non convexes*

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7			0.95
Lien simple	6	5		0.95
Lien simple	5	5	0.036	0.91
Lien simple	4	4	0.0001	0.1
Lien simple	3	3		0.60
Lien simple	2	2		0.87
Lien simple	1			
Lien complet	7			0.75
Lien complet	6	6	0.0006	0.85
Lien complet	5	5	0.000085	0.71
Lien complet	4	4		0.81
Lien complet	3	3		0.77
Lien complet	2	2		0.91
Lien complet	1			
Ward	7			0.92
Ward	6	6	0.004	0.85
Ward	5	5	0.004	0.71
Ward	4	4		0.81
Ward	3	3		0.77
Ward	2	2		0.91
Ward	1			
Centroïde	7			0.92
Centroïde	6	6	0.00005	0.86
Centroïde	5	5	0.0000005	0.79
Centroïde	4	4		0.66
Centroïde	3	3		0.76
Centroïde	2	2		0.82
Centroïde	1			

TAB. 3.7 – *Résultats obtenus - Classes non convexes en 3 dimensions*

3.4 Conclusion

Nous avons donc appliqué le gap test et le test des hypervolumes à des données classiques. Après l'analyse des résultats obtenus, la première remarque que l'on peut formuler est que le test des hypervolumes donne le plus souvent de meilleurs résultats.

On a ensuite remarqué que les biais des méthodes de classification influencent les résultats des deux tests pour certains jeux de données. Pour des données présentant des classes parallèles par exemple, seule la suite de partitions données par la méthode du lien simple permettait de retrouver le bon nombre de classes. Pour d'autres données, le gap test retrouvait le nombre de classes naturelles mais la partition donnée par certaines méthodes n'était pas la partition naturelle des données. C'est pourquoi il est important de connaître les biais des méthodes de classification.

Pour certains exemples, on ne sait pas expliquer les mauvais résultats du gap test. Cependant, pour la plupart des jeux de données, le gap test retrouve le bon nombre de classes pour la suite de partitions données par la méthode du lien simple, à l'exception des données de Ruspini, des données présentant des classes non séparables par un hyperplan et des données présentant des classes non convexes.

Chapitre 4

Les données symboliques

4.1 Introduction

Ce chapitre se base sur le livre de Bock et Diday [1].

Dans beaucoup de domaines d'activités, on recueille aujourd'hui de plus en plus de données. Il devient donc important de pouvoir les résumer en extrayant les informations principales. Un des outils que l'on peut utiliser sont les données symboliques.

Mais ce n'est pas leur seule application, généralement, les variables que l'on mesure sur des individus ne fournissent qu'une seule valeur par individus. On obtient de la sorte un jeu de données classiques. De nos jours, cependant, les données récoltées ont une structure plus complexe que les données classiques.

En effet, si on demande à un étudiant le nombre d'heures qu'il a travaillé par jour pendant la semaine, ce nombre peut varier entre, par exemple, une demi-heure et quatre heures. La réponse n'est pas une valeur unique mais on peut la représenter par un intervalle, c'est un des types possibles de variables symboliques que l'on peut rencontrer.

L'objectif de la classification automatique étant de rechercher une structure en classes parmi les individus, on pourrait aussi vouloir résumer les données récoltées sur des individus, par la mesure de variables sur les classes. On décrirait ainsi les classes grâce à la description des individus de la classe, on utilise aussi pour cela des variables symboliques.

4.2 Les données classiques

Avant de décrire les données symboliques, nous allons rappeler quelques notions sur les données classiques.

On dispose d'un ensemble $\Omega = \{1,2,\ldots,n\}$ d'individus simples, sur lesquels on mesure la valeur de p variables $Y_1,Y_2,\ldots,Y_p$. Chaque variable Y_j prend ses valeurs dans un certain ensemble $\mathcal{Y}_j$ appelé l'espace d'observations de Y_j. Pour chaque individu $k \in \Omega$, Y_j réalise une seule valeur $Y_j(k)$ de $\mathcal{Y}_j$.

$$\begin{aligned} Y_j : \quad & \Omega \to \mathcal{Y}_j \\ & k \rightsquigarrow Y_j(k) \equiv x_{kj} \end{aligned}$$

Toutes les valeurs x_{kj} sont rassemblées dans la matrice de données X. Chaque cellule contient un seul élément x_{kj} et chaque ligne décrit un individu.

On peut distinguer deux types de variables, les variables quantitatives et les variables qualitatives.

4.2.1 Les variables quantitatives

Une variable Y est une variable quantitative continue si $\mathcal{Y} = \mathbb{R}$ ou $\mathcal{Y} = \mathbb{R}^+$ ou $\mathcal{Y} = [a,b]$. Une variable quantitative est discrète si $\mathcal{Y} = \{\xi_1,\xi_2,\ldots\xi_M\} \subset \mathbb{R}$ ou $\mathcal{Y} = \{\xi_1,\xi_2,\ldots\} \subset \mathbb{R}$. Pour ces variables une mesure de proximité $\delta(x,y)$ entre paires d'éléments, est la distance euclidienne.

4.2.2 Les variables qualitatives

Une variable Y est dite qualitative si $\#\mathcal{Y} < \infty$ et si les éléments de $\mathcal{Y}$ sont des catégories. On distingue les variables nominales pour lesquelles il n'y a pas de structure dans $\mathcal{Y}$ et la seule chose qu'on peut dire sur une paire d'éléments x,y de $\mathcal{Y}$ c'est si $x = y$ ou $x \neq y$. Dans ce cas une mesure de proximité entre paires d'éléments peut être: $\delta(x,y) = 1$ si $x = y$ et 0 sinon. Si $\mathcal{Y} = \{0,1\}$ la variable nominale est appelée binaire.

Une variable qualitative est ordinale si $\mathcal{Y}$ est muni d'un ordre total $\prec$ tel que $\forall\ a,\ b \in \mathcal{Y}$, $a \prec b$ ou $b \prec a$. On parle de variables ordinales généralisées si on a un ordre partiel $\prec$, c'est-à-dire que toutes les paires de $\mathcal{Y}$ ne sont pas comparables. Le système de paires ordonnées peut être mis sous la forme d'un diagramme hiérarchique, ou réseau tel que $\forall\ a,\ b \in \mathcal{Y}$, $b \prec a$ si

et seulement si il existe une suite d'arêtes connexes reliant a à b. Le cas particulier de ces variables ordinales généralisées se produit quand $\mathcal{Y}$ est donné par un arbre hiérarchique, c'est-à-dire que les catégories sont ordonnées dans un arbre hiérarchique, cette variable est appelée une variable taxonomique.

4.2.3 Les variables dépendantes

La dépendance logique

Une dépendance logique existe entre les deux variables Y et Z si les valeurs de la variable Z dépendent logiquement ou fonctionnellement des valeurs de la variable Y.

Par exemple, si $Y :=$ poids et $Z :=$ taille on peut définir une dépendance logique par: si $Y \leq 50$ alors $Z \leq 185$.

La dépendance hiérarchique

La variable Z est dépendante de façon hiérarchique à la variable Y si l'espace d'observations $\mathcal{Z}$ de Z dépend de la valeur $y \in \mathcal{Y}$ prise par Y.

Par exemple, si $Y :=$ possède une voiture et $Z :=$ nombre d'accidents, on a $\mathcal{Y} = \{0,1\}$ (0 pour ne possède pas de voiture et 1 pour possède une voiture). $\mathcal{Z}_1 = \{0,1,2,\dots\}$, $\mathcal{Z}_0 = \{NA\}$ où NA signifie non applicable. $\mathcal{Z} = \{NA,0,1,2,\dots\} = \mathcal{Z}_0 \bigcup \mathcal{Z}_1$.

La dépendance stochastique

Les variables aléatoires $Y_1,\dots,Y_p$ sont stochastiquement indépendantes si et seulement si $P(Y_1 \in B_1,\dots,Y_p \in B_p) = \prod_{j=1}^{p} P(Y_j \in B_j)$ où B_j est un ensemble mesurable $B_j \subset \mathcal{Y}_j$.

4.3 Les données symboliques

Dans la section précédente, les individus étaient décrits par des variables qui prenaient une valeur unique. Cependant, le monde réel est trop complexe pour être décrit par ces variables; c'est pourquoi on introduit les variables

symboliques.

On distinguera

- $E = \Omega = \{1,2,\ldots,n\}$, les individus, les objets du premier ordre;

- $E = \{C_1,C_2,\ldots\}$; $C_j \subset \Omega$, les classes d'individus, les objets du second ordre.

4.3.1 Quelques exemples de données symboliques

Pour des objets du premier ordre

Soit $\Omega = \{1,2,\ldots,n\}$, on peut considérer la variable suivante, $Y :=$ "Temps passé à écouter la radio par jour".

On peut avoir par exemple, $Y(k) = [0,3]$, Y est une variable quantitative de type intervalle. Le temps passé à écouter la radio varie de 0 à 3 heures par exemple. Ou encore, $Y(k) = ((0,0.5),(1,0.4),(2,0.05),(3,0.05))$. Y est une variable quantitative modale. Le temps passé à écouter la radio est de 0 heure avec une probabilité de 0.5, de 1 heure avec une probabilité de 0.4, de 2 heures avec une probabilité de 0.05 et de 3 heures avec une probabilité de 0.05.

Pour des objets du second ordre

Soit $E = \{$Villes d'un pays$\}$ et la variable Y suivante, $Y :=$ "Secteur industriel présent".

On aura par exemple $Y(k) =\{$sidérurgie, électronique, textile$\}$, Y est une variable nominale multivaluée, on peut aussi avoir $Y(k) =$((sidérurgie,0.5), (électronique,0.45), (textile,0.05)) et Y est une variable nominale modale. Dans ce cas, $Y(k)$ peut être un histogramme reprenant les différents secteurs et les probabilités correspondantes.

Dans les sections suivantes nous définirons plus précisément quelques types de variables symboliques comme les variables multivaluées, intervalles et modales.

4.3.2 Variables multivaluées et intervalles

Définitions

Soit $Y : E \rightarrow \mathcal{Y}$ telle que $Y(k) \in \mathcal{P}(\mathcal{Y})$. Si $|Y(k)| = 1$, $\forall\ k \in E$, Y est une variable univaluée classique.

- Y est multivaluée si $|Y(k)| < \infty$, $\forall k \in E$.
- Y est multivaluée catégorique si $|Y(k)| < \infty$, $\forall k \in E$ et si $\mathcal{Y}$ a un nombre fini de catégories.
- Y est multivaluée quantitative si les valeurs $Y(k)$ sont des ensembles finis de réels, c'est-à-dire $Y(k) \subset \mathbb{R}$ et $|Y(k)| < \infty$, $\forall k \in E$.
- Y est une variable intervalle si $\forall k \in E$, $Y(k)$ est un intervalle de $\mathbb{R}$.

Exemples

Variable multivaluée catégorique

Soit

$E =$ {Pays}

$\mathcal{Y} =$ {Les journaux possibles}
$=$ {Le Soir, Le Monde, Le Figaro, Le New-York Times,
Le Times, La Libre Belgique, Le Sun,
Le Washington Post, Le Chicago Tribune, ...}

$Y(k)$:= "Les journaux propres au pays k"

On obtient,

k	$Y(k)$
France	{Le Monde, Le Figaro}
Belgique	{La Libre Belgique, Le Soir}
Royaume-Uni	{Le Sun, Le Times}
Etats-Unis	{Le New-York Times, Le Washington Post, Le Chicago Tribune}

Variable multivaluée quantitative

Soit

E= {Pays}

Y:= "Nombre d'habitants dans les cinq plus grandes villes"

Y peut prendre, par exemple comme valeur, $Y(k)$ = {40000, 70000, 20000, 35000, 65000}. $Y(k)$ est un sous ensemble fini de $\mathcal{Y} = \mathbb{R}^+$.

Variable intervalle

Soit

E = {Automobilistes}

Y:= "Trajet parcouru par jour (en km)"

$\mathcal{Y}$ = $\mathcal{I}$, l'ensemble des intervalles de $\mathbb{R}$

On aura, par exemple $Y(k)$= [0,20], $Y(l)$ = [35,85], ...

4.3.3 Variables multivaluées et intervalles par agrégation

Il est possible de définir une variable symbolique pour des objets du second ordre à partir de valeurs prises par des variables classiques sur des objets du premier ordre. C'est ce qu'on appelle des variables symboliques par agrégation.

Ainsi, si on considère $\Omega = \{1, \ldots, n\}$ les objets du premier ordre et $\widetilde{Y}$ une variable univaluée classique sur Ω à valeur dans $\mathcal{Y}$, on peut aussi considérer $E = \{C_1, \ldots, C_m\}$ des objets du second ordre, avec $C_i \subset \Omega$, sur lesquels on va définir Y une variable symbolique, à partir des valeurs prises par $\widetilde{Y}$ sur les objets du premier ordre.

Exemple

- Soit Ω = {Elèves d'une école}, les objets du premier ordre sur lesquels on mesure la valeur de la variable classique $\widetilde{Y}$ telle que $\widetilde{Y}(k) :=$ moyenne de l'élève k, $k \in \Omega$.

- On considère $E = \{C_1, \ldots, C_m\}$={m classes d'une école} les objets du second ordre.

On peut alors par exemple construire une variable symbolique Y telle que $Y(C_i)$={7.4; 8.5; 6.7; 9.1; 5.4; 7.6} pour une classe de 6 élèves. Y est une variable multivaluée quantitative obtenue par agrégation. $Y(C_i)$ est une description de la classe C_i.

On peut aussi obtenir une variable intervalle par agrégation, si on prend $Y(C_i)$=[min,max]=[α,β]. Où $\alpha = \min_{w \in C_i} \{\widetilde{Y}(w)\}$ et $\beta = \max_{w \in C_i} \{\widetilde{Y}(w)\}$.

On peut également décrire une classe par $\overline{x}_{C_i}$ et $s^2_{C_i}$ mais dans ce cas on ne sait pas décider de l'appartenance d'un nouvel élève.

4.3.4 Variable modale

Définition

- Y est une variable modale sur $E = \{a, b, \ldots\}$ si c'est une variable multivaluée pour laquelle on a :

  - $\forall a \in E$, $Y(a) \subset \mathcal{Y}$

  - $\forall y \in Y(a)$, $w(z)$ indique la fréquence avec laquelle y se produit pour l'objet a.

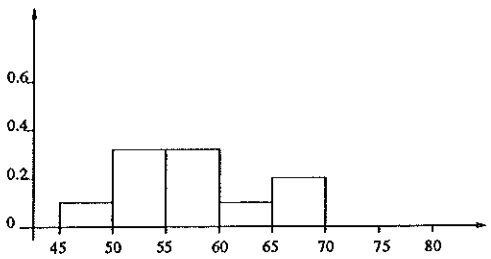
La variable modale peut se représenter par un histogramme.

Exemple

- Soit $\Omega = \{1,\ldots,100\}$ des étudiants et $\widetilde{Y}(k)$:= poids de l'étudiant k
- Considérons $E = \{C_1,\ldots,C_m\}$ et une classe quelconque C=$\{1,\ldots,10\}$ de 10 étudiants

Pour ces étudiants, $\widetilde{Y}$ peut prendre les valeurs suivantes,
$\widetilde{Y}$ = 46, 55, 51, 53, 60, 60, 62, 66, 58, 70.

La variable symbolique modale Y qui décrit la classe C peut avoir la réalisation suivante:



Ce qui veut dire qu'un étudiant sur les dix pèse entre 46 et 50 kilogrammes, trois étudiants pèsent entre 51 et 55 kilogrammes, ...

4.3.5 Tableau de données symboliques

Comme pour les données classiques, on peut rassembler les valeurs prises par des variables symboliques pour différents individus dans un tableau appelé tableau de données symboliques.

- Soit $E = \{1,2,\ldots,N\}$ l'ensemble de base et $u \in E$ un objet quelconque.

  On aura par exemple :

  - $E = \Omega = \{1,2,\ldots,n\}$, $N = n$
  - $E \subset \Omega$, un échantillon de Ω, $N \leq n$
  - $E = \{C_1,\ldots,C_m\}$ classes $\subseteq \Omega$, des objets du second ordre, $N = m$

- Considérons aussi $Y_1,\ldots,Y_p$, p variables symboliques

- On note par $x_u = X(u) := (Y_1(u),\ldots,Y_p(u))'$ le vecteur des variables symboliques pour $u \in E$. $X(u) = (\xi_{u1},\ldots,\xi_{up})'$ où $\xi_{uj} = Y_j(u)$

- Le tableau de données symboliques correspondant est donné par :

$$X = \begin{pmatrix} x_1' \\ \vdots \\ x_N' \end{pmatrix} = \begin{pmatrix} \xi_{11} & \cdots & \xi_{1p} \\ \vdots & \ddots & \vdots \\ \xi_{N1} & \cdots & \xi_{Np} \end{pmatrix} = (\xi_{uj})_{N\times p}$$

Une ligne de la matrice est une description symbolique de l'objet correspondant. Remarquons qu'une cellule de la matrice peut très bien contenir un histogramme si la variable symbolique considérée est modale.

Exemple

$$X = \left(\begin{array}{c|c|c} [20,60] & (A,0.4;B,0.3;C,0.3) & \{Le\ Soir,\ La\ Libre\} \\ [40,60] & (A,0.1;B,0.5;C,0.4) & \{Ch.\ Tribune,\ NY\ Times\} \\ [4,8] & (A,0.3;B,0.5;C,0.2) & \{Times\} \\ [12,15] & (A,0.3;B,0.4;C,0.3) & \{Le\ Figaro,\ Le\ Monde\} \end{array}\right)$$

La première colonne de la matrice correspond aux valeurs de la variable Y_1 de type intervalle. La deuxième variable Y_2, dont les valeurs sont dans la deuxième colonne, est une variable modale. Chaque cellule de cette colonne peut être remplacée par un histogramme. La troisième colonne correspond à une variable catégorique multivaluée Y_3.

On a vu qu'on pouvait construire des variables symboliques par agrégation grâce à des variables univaluées classiques, cela permet aussi de réduire la taille de certains tableaux. En effet, si on dispose d'un tableau, comprenant la description de plusieurs individus par des variables classiques, on peut résumer l'information de ces données en utilisant des variables symboliques sur des objets du second ordre qui sont des classes de ces individus. Par exemple, si les différents individus sont des voitures sur lesquelles on a récolté certaines informations, on peut décider de considérer des classes de voitures, formées par des voitures d'un même pays d'origine et de récolter les informations concernant les classes et non plus les voitures.

4.4 Les objets symboliques

Comme on vient de le voir, grâce aux variables symboliques définies par agrégation, on peut résumer l'information récoltée pour plusieurs individus par une information sur des classes de ces individus. On peut aussi résumer l'information ou bien réduire le nombre d'individus à considérer en choisissant uniquement ceux qui nous intéressent. On définit pour cela des objets symboliques. Le problème de base est le suivant: trouver les objets $u \in E$ qui remplissent certaines conditions par rapport aux variables symboliques $Y_1, \ldots, Y_p$. Toutes ces conditions sont reprises dans une QUERY, une requête.

Si on dispose de quatre variables binaires, on peut formuler par exemple, la recherche (QUERY) suivante: $q = [Y_1 = 1] \wedge [Y_2 = 1] \wedge [Y_4 = 0]$. L'idée "tous les individus qui satisfont q" est un concept appelé objet symbolique et l'ensemble de tous les individus qui vérifient q est appelé l'extension dans

E de la requête q ou de l'objet symbolique sous-jacent.

Considérons le tableau de données (Tableau 4.1) qui représente sept individus suivant quatre variables. Homme (codé 0) ou femme (codé 1), marié (codé 1) ou pas (codé 0), travaillant (codé 1) ou pas (codé 0), ayant des enfants (codé 1) ou pas (codé 0).

u	Individus Nom	Sexe Y_1	Marié Y_2	Emploi Y_3	Enfant Y_4
1	A	1	1	1	0
2	B	1	1	1	1
3	C	1	1	1	0
4	D	0	1	1	1
5	E	0	1	1	1
6	F	0	0	1	1
7	G	0	0	0	1

TAB. 4.1 – *Tableau de données*

La demande q formulée précédemment définit pour cet exemple, l'objet symbolique "Femme mariée n'ayant pas d'enfant" et a comme extension les individus A et C.

4.5 La détermination du nombre de classes pour des données symboliques de type intervalle

Dans ce travail, nous allons appliquer le test des hypervolumes et le gap test à des données symboliques de type intervalle. Il nous faut auparavant décrire comment des données de type intervalle peuvent se représenter dans l'espace et comment on pourra calculer l'enveloppe convexe d'un tel ensemble de données.

On dispose d'un ensemble d'objets, d'individus $\Omega = \{1,2,\ldots,n\}$ et de p variables de type intervalle $Y_1,\ldots,Y_p$ telles que:

$$\begin{aligned} Y_j : \ & \Omega \to \mathcal{I} \\ & k \rightsquigarrow Y_j(k) = [\alpha,\beta] \subset \mathbb{R} \end{aligned}$$

où $\mathcal{I}$ est l'ensemble de tous les intervalles fermés et bornés de $\mathbb{R}$. La matrice de données est $X = (\xi_{kj})_{n\times p}$ où $\xi_{kj} = Y_j(k)$ est un intervalle.

Représentation géométrique

On représente un individu k par un hyperrectangle dans un espace euclidien p-dimensionnel.

Si on a par exemple un objet sur lequel on mesure la valeur de deux variables Y_1 et Y_2 de type intervalle, on peut représenter l'objet dans un espace de dimension 2 (Fig. 4.1).

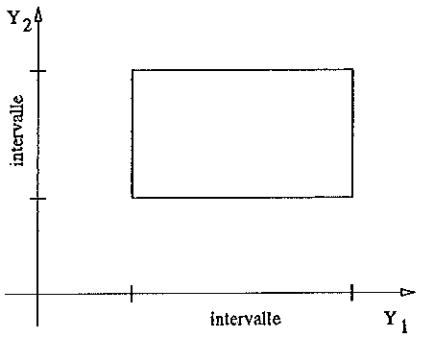


FIG. 4.1 –

Notion d'enveloppe convexe pour des variables intervalles

Rappelons que dans le test des hypervolumes tout comme dans le gap test, il nous faut calculer les enveloppes convexes des classes. Dans le cas de données de type intervalle cela pose problème. Qu'est-ce que l'enveloppe convexe pour des objets de type intervalle?

Nous avons vu que tout objet pouvait être représenté par un hyperrectangle dans $\mathbb{R}^p$. Pour obtenir une notion d'enveloppe convexe pour ces objets, on simule un processus de Poisson homogène dans les hyperrectangles représentant les objets.

On définit alors l'enveloppe convexe d'une classe comme l'enveloppe convexe de tous les points simulés dans les hyperrectangles représentant les éléments de cette classe. On se ramène donc à un problème avec des données quantitatives classiques.

C'est ce que nous allons faire pratiquement pour appliquer le test des hypervolumes et le gap test à des données de type intervalle. Nous simulerons un processus de Poisson homogène dans chaque hyperrectangle représentant les objets symboliques. Nous appliquerons ensuite les différentes méthodes de classification pour obtenir une partition en k classes et nous chercherons finalement le nombre de classes naturelles en appliquant le test des hypervolumes et le gap test aux différentes partitions obtenues. Mis à part la simulation du processus de Poisson, on travaille comme avec des données classiques.

Remarquons qu'il existe des méthodes capables de traiter telles quelles des données symboliques de type intervalle et autres types. La méthode divisive de Chavent [2] par exemple, qui définit une mesure de dissimilarité pour ces variables et généralise aux données symboliques le critère de la variance.

Chapitre 5

Résultats du gap test et du test des hypervolumes appliqués à des données symboliques de type intervalle

5.1 Introduction

Nous avons appliqué le gap test et le test des hypervolumes à des données symboliques de type intervalle. Nous avons traité deux jeux de données. Comme nous l'avons déjà expliqué dans le chapitre précédent, avant de traiter les données, nous avons simulé un processus de Poisson homogène dans tous les hyperrectangles représentant les objets symboliques.

5.2 Données symboliques de type intervalle, deux variables

Ce jeu de données est constitué de 16 individus sur lesquels on a mesuré la valeur de 2 variables de type intervalle. Chaque individu peut être représenté par un hyperrectangle dans le plan. Dans chacun de ces hyperrectangles, nous avons simulé des points selon un processus de Poisson homogène. Nous en avons simulés cinq dans chaque hyperrectangle. En effet, il nous faut obtenir suffisamment de points pour pouvoir appliquer les différentes méthodes de classification.

Simulation d'un processus de Poisson pour des données de type intervalle

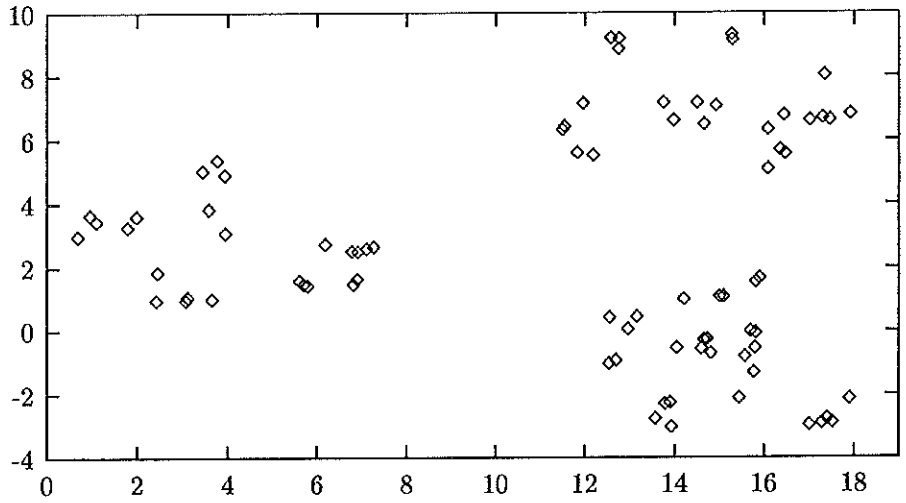


FIG. 5.1 –

On peut donc considérer le jeu de données comme un jeu de données classiques constitué de 80 individus sur lesquels on a mesuré deux variables, celui-ci est représenté sur le graphe suivant (Fig. 5.1).

On applique ensuite les différentes méthodes de classification à ces données pour obtenir les partitions en classes avec lesquelles on appliquera le gap test et le test des hypervolumes.

Description des résultats

Les données présentent, comme le montre le graphe (Fig. 5.1), trois classes. Les résultats obtenus sont présentés dans le tableau suivant (Tableau 5.1).

Le test des hypervolumes retrouve les trois classes pour toutes les méthodes de classification. Remarquons que la partition en trois classes données par toutes les méthodes de classification est la partition naturelle des données en trois classes.

Le gap test retrouve la partition en trois classes uniquement pour la méthode du lien simple. Pour les autres méthodes, le gap test accepte les

partitions en 6, 5, 4, 3 et 2 classes. Pour ces méthodes on peut conclure uniquement qu'il y a au moins 6 classes dans les données. Nous avons également essayé d'ajuster la valeur de α afin d'obtenir de meilleurs résultats. Si cette valeur est fixée à un maximum de 0.0025, on obtient, pour toutes les méthodes de classification, le bon nombre de classes naturelles. On peut dire que les résultats donnés par le test des hypervolumes sont meilleurs que ceux donnés par le gap test.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test de hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7			0.86
Lien simple	6	5		0.88
Lien simple	5	4		0.86
Lien simple	4	3		0.96
Lien simple	3	3		0.65
Lien simple	2	2		0.58
Lien simple	1			
Lien complet	7			0.84
Lien complet	6	6	0.02	0.84
Lien complet	5	5	0.02	0.86
Lien complet	4	4	0.0025	0.86
Lien complet	3	3		0.65
Lien complet	2	2		0.58
Lien complet	1			
Ward	7			0.89
Ward	6	6	0.005	0.83
Ward	5	5	0.005	0.83
Ward	4	4	0.0025	0.86
Ward	3	3		0.65
Ward	2	2		0.58
Ward	1			
Centroïde	7			0.86
Centroïde	6	6	0.005	0.81
Centroïde	5	5	0.0025	0.86
Centroïde	4	4	0.0025	0.86
Centroïde	3	3		0.65
Centroïde	2	2		0.58
Centroïde	1			

TAB. 5.1 – *Résultats obtenus - Données symboliques de type intervalle*

5.3 Données symboliques de type intervalle, quatre variables

Il s'agit en fait d'un jeu de données réelles. Ces données représentent 8 individus sur lesquels on mesure 4 variables de type intervalle. Ces données sont présentées dans le tableau suivant (Tableau 5.2).

Sample	Specific Gravity	Freezing point	Iodine Value	Saponification Value
linseed oil	[0,930;0,935]	[-27;-18]	[170;204]	[118;196]
perilla oil	[0,930;0,937]	[-5;-4]	[192;208]	[188;197]
cottonseed oil	[0,916;0,918]	[-6;-1]	[99;113]	[189;198]
sesam oil	[0,920;0,926]	[-6;-4]	[104;116]	[187;193]
camelia oil	[0,916;0,917]	[-21;-15]	[80;82]	[189;193]
olive oil	[0,914;0,919]	[0;6]	[79;90]	[187;196]
beef tallow	[0,860;0,870]	[30;38]	[40;48]	[190;199]
hog fat	[0,858;0,864]	[22;32]	[53;77]	[190;192]

TAB. 5.2 – *Données réelles*

De la même manière que pour le jeu de données précédent, on va simuler des points selon un processus de Poisson homogène dans les hyperrectangles représentant les objets symboliques. Pour ces données nous en avons simulé 10 dans chaque hyperrectangle afin d'obtenir assez de points pour l'application des méthodes de classification. Après la simulation, on dispose donc de 80 points dans un espace de dimension 4.

On applique à ces données les différentes méthodes de classification et ensuite le gap test et le test de hypervolumes aux partitions obtenues.

Description des résultats

Ces données réelles présentent normalement 3 ou 4 classes. Le test des hypervolumes trouve une partition en 4 classes pour les méthodes du lien simple et du lien complet et une partition en 3 classes pour les deux autres méthodes.

Le gap test accepte pour la première fois la partition en 4 classes uniquement pour la méthode du lien simple. Pour les autres méthodes, il accepte

toutes les partitions en 7, 6, 5, 4, 3 et 2 classes. Les résultats que nous avons obtenus sont représentés dans le tableau suivant (Tableau 5.3). Dans cet exemple, si on ajuste la valeur de α à 0.004, on obtient les mêmes résultats que le test des hypervolumes, c'est-à-dire un nombre de 4 classes naturelles pour la méthode du lien complet et 3 classes pour la méthode du centroïde. Par contre, pour la méthode de Ward, les valeurs de α doivent être extrêmement petites pour obtenir de bons résultats. On peut donc dire encore une fois que les résultats donnés par le test des hypervolumes sont meilleurs que ceux donnés par le gap test.

Méthode de classification	Nombre de classes dans la partition (k)	Résultats du gap test		Résultats du test des hypervolumes
		$\alpha = 0.05$	Valeur ajustée de α	
Lien simple	7	6		0.95
Lien simple	6	5		0.61
Lien simple	5	4		0.97
Lien simple	4	4		0.09
Lien simple	3	3		0.88
Lien simple	2	2		0.11
Lien simple	1			
Lien complet	7	7	0.035	0.53
Lien complet	6	6	0.035	0.53
Lien complet	5	5	0.035	0.71
Lien complet	4	4		0.63
Lien complet	3	3		0.20
Lien complet	2	2		0.11
Lien complet	1			
Ward	7	7	0.00001	0.88
Ward	6	6	0.000008	0.45
Ward	5	5	0.000004	0.26
Ward	4	4		0.78
Ward	3	3		0.20
Ward	2	2		0.11
Ward	1			
Centroïde	7	7	0.01	0.38
Centroïde	6	6	0.01	0.44
Centroïde	5	5	0.004	0.66
Centroïde	4	4	0.004	0.63
Centroïde	3	3		0.20
Centroïde	2	2		0.11
Centroïde	1			

TAB. 5.3 – *Résultats obtenus - Données de type intervalle, 4 dimensions*

Conclusion

L'objectif de ce mémoire est tout d'abord d'implémenter la méthode du gap test pour pouvoir par la suite, l'appliquer à des données classiques et symboliques de type intervalle. Nous avons également appliqué le test des hypervolumes à ces mêmes données afin de comparer les résultats obtenus par les deux tests. Les programmes du gap test et du test des hypervolumes sont rédigés en Fortran 77.

Avant de présenter les conclusions relatives à cette comparaison, rappelons que les deux méthodes de détermination du nombre de classes utilisées se basent sur des partitions des données obtenues par des méthodes de classification. C'est pourquoi il est important de connaître les biais de ces méthodes. En effet, nous avons constaté que les tests ne déterminaient pas le nombre correct de classes lorsqu'ils étaient associés à des méthodes de classification ne retrouvant pas les classes naturelles. Par ailleurs, même s'ils retrouvaient dans ce cas le nombre correct de classes, les classes proposées par la partition ne reflétaient pas la véritable structure des données.

La comparaison des résultats est présentée dans les tableaux suivants (Tableau 5.4 et Tableau 5.5). Pour chaque jeu de données nous avons récolté les résultats obtenus par l'application des deux tests aux suites de partitions données par les différentes méthodes de classification. Une $\times$ signifie que le test retrouve le bon nombre de classes pour la partition naturelle des données. Un $-$ signifie que le test ne trouve pas le bon nombre de classes naturelles.

Une première lecture des résultats obtenus pour les données classiques, nous permet de dire que dans la plupart des cas, le test de hypervolumes retrouve le bon nombre de classes. Il faut bien sûr tenir compte des biais des différentes méthodes de classification et des hypothèses du test. Ainsi, pour des classes parallèles ou allongées, le test des hypervolumes détermi-

nera le bon nombre de classes uniquement pour les partitions données par la méthode du lien simple. De même, ce test ne nous permet pas de conclure pour des données présentant des classes qui ne respectent pas l'hypothèse de convexité des classes naturelles, ni pour des données présentant des classes non séparables par un hyperplan.

Les résultats obtenus par le gap test sont par contre moins bons. De plus, dans le cas où il ne retrouve pas le bon nombre de classes, il faut fixer le seuil α à des valeurs extrêmement petites pour obtenir de meilleurs résultats. De même que pour le test des hypervolumes, le gap test retrouve le bon nombre de classes pour des données présentant des classes parallèles uniquement pour les partitions obtenues par la méthode du lien simple étant donné les biais des autres méthodes. Le test retrouve aussi le bon nombre de classes pour des données présentant deux groupes de tailles inégales. Pour les autres données, on peut remarquer que le gap test retrouve le bon nombre de classes pour les partitions généralement obtenues par la méthode du lien simple excepté pour les données de Ruspini, les données présentant des classes non séparables par un hyperplan et les données présentant des classes non convexes.

Nous avons également appliqué ces deux tests à des données symboliques de type intervalle. Pour traiter ces données, nous avons simulé dans chaque hyperrectangle représentant les objets symboliques un processus de Poisson homogène. Nous obtenons de la sorte un jeu de données classiques. Le premier jeu de données nous amène aux mêmes conclusions que pour les données classiques. Le deuxième jeu provenait de données réelles en 4 dimensions. Le test des hypervolumes retrouve les bonnes partitions pour toutes les méthodes de classification et le gap test donne le bon résultat pour la suite de partitions obtenue par la méthode du lien simple. Pour ces exemples aussi le seuil α du gap test doit être fixé à des valeurs extrêmement petites pour obtenir de meilleurs résultats.

Remarquons qu'il serait intéressant, d'analyser les partitions obtenues par les différentes méthodes de classification afin de déterminer pourquoi le gap test donne le plus souvent de bons résultats pour les partitions de la méthode du lien simple.

Enfin, rappelons qu'il est toujours plus sage de se baser sur plusieurs méthodes de détermination du nombre de classes avant de conclure sur ce nombre.

Types de données	Méthodes de classification	Résultats du gap test	Résultats du test des hyper-volumes
Tailles inégales dim 2	Lien simple	×	×
	Lien complet	×	×
	Ward	×	×
	Centroïde	×	×
Deux classes parallèles dim 2	Lien simple	×	×
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–
Trois classes parallèles dim 2	Lien simple	×	×
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–
Deux groupes éloignés dim 3	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	–	×
Deux groupes proches dim 3	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	×	×
Trois groupes bien séparés dim 2	Lien simple	×	×
	Lien complet	×	×
	Ward	–	×
	Centroïde	×	×
Deux classes bien séparées dim 2	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	×	×
Trois classes allongées dim 2	Lien simple	×	×
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–
Données non séparables dim 3	Lien simple	–	–
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–

TAB. 5.4 – *Comparaison des résultats*

Types de données	Méthodes de classification	Résultats du gap test	Résultats du test des hyper-volumes
Données sans structure dim 2	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	–	×
Données de Ruspini	Lien simple	–	×
	Lien complet	–	–
	Ward	–	×
	Centroïde	–	×
Classes non convexes dim 2	Lien simple	–	–
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–
Classes non convexes dim 3	Lien simple	–	×
	Lien complet	–	–
	Ward	–	–
	Centroïde	–	–
Données symboliques 1er exemple	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	–	×
Données symboliques 2ème exemple données réelles	Lien simple	×	×
	Lien complet	–	×
	Ward	–	×
	Centroïde	–	×

TAB. 5.5 – *Comparaison des résultats*

Bibliographie

[1] H-H. Bock et E. Diday. *Analysis of symbolic data.* Springer, Berlin, 2000.

[2] H-H. Bock et E. Diday. *Analysis of symbolic data*, chapitre 11, pages 299-311. Springer, Berlin, 2000. Criterion-based divisive clustering for symbolic data.

[3] J.P Chandon et S. Pinson. *Analyse typologique : théories et applications.* Masson, Paris, 1981.

[4] A. Hardy. *Une nouvelle approche des problèmes de classification automatique. Un modèle - Un nouveau critère - Des algorithmes - Des applications.* PhD thesis, FUNDP, 1983.

[5] A. Hardy. An examination of procedures for determining the number of clusters in a data set. In *: E.Diday et al. (eds)*, pages 178-185. Springer, Berlin, 1994.

[6] A. Hardy. *Syllabus de cours, aspect statistique de la classification*, 2001-2002.

[7] T. Kubushishi. *On some applications of the point process theory in cluster analysis and pattern recognition.* PhD thesis, FUNDP, 1996.

[8] J-P. Rasson et T. Kubushishi. The gap test: an optimal method for determining the number of natural classes in cluster analysis. In *: E.Diday et al. (eds)*, pages 186-193. Springer, Berlin, 1994.

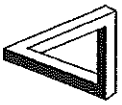
[9] B.D. Ripley et J.P. Rasson. Finding the edge of a Poisson forest. *Journal of Applied Probability*, 14, 1977.

[10] SAS/STAT. *User's Guide.* SAS institute Inc, NC, USA, 1990.

FUNDP
Faculté des Sciences
Département de Mathématique

Rempart de la Vierge, 8
B-5000 Namur Belgique

Comparaison entre le gap test et le test des hypervolumes en classification (Annexes)



Mémoire présenté pour l'obtention
du grade de
Licencié en Sciences Mathématiques
par

Promoteur : A. Hardy

Catherine BEAUTHIER

Année académique 2001-2002

Table des matières

A Programme du gap test 2

B Programme du test des hypervolumes 62

Annexe A

Programme du gap test

```
c     ************************************************************
c     *                                                          *
c     *                  Programme gaptest                       *
c     *                                                          *
c     ************************************************************
c
c
c     A partir d'une partition en k classes des données triée par
c     classes et obtenue par sas, le programme permet d'effectuer,
c     pour chaque paire de classes le test d'hypothèses suivant:
c
c     H 0 : les deux classes doivent être réunies
c
c     H 1 : les deux classes doivent rester séparées
c
c     Pour cela on calcule l'enveloppe de chaque classes,
c     l'enveloppe convexe des deux classes réunies et
c     on calcule ensuite les transformations T(i,j) en
c     utilisant la constante kappa calculée au préalable.
c
c     Si le minimum de la matrice T ainsi construite est plus petit
c     que la valeur de talpha, alors les deux classes doivent être
c     réunies.
c
c     Le fichier contenant la partition doit avoir la forme suivante :
c
c     Numéro de          Coordonnées    Numéro de
c     l'observation        X    Y       la classe
c
c     OBS 1              0.99  0.45        1
c     OBS 2              0.45  0.34        1
c     OBS 3              0.89  0.65        1
c     OBS 4              1.45  1.84        2
```

```
c     OBS 5              1.99  1.45        2
c     OBS 6              1.56  1.67        2
c     OBS 7              1.59  1.75        2
c     OBS 8              1.75  1.84        2
c
c
c
      program gaptest

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'


      double precision PI

      INTEGER k
      parameter (k=4)

      integer dim
      parameter (dim=4)

      integer n
      parameter (n=80)

      real alpha
      parameter (alpha=0.05)

      real talpha

      integer ncli,nclj

      double precision volume, vol(k)
      double precision MD

      real x(dim,n), c(dimmax,npmax), d(dimmax,npmax)
      double precision T(dimmax,npmax)

      integer cl(n), nb(0:k)
      integer i,j,f,y,z,g
      real m
      real min

      integer nom(n)
```

```
      double precision kappa


c     Description des variables de gaptest
c     ------------------------------------
c
c     PI: contiendra la constante pi utilisée dans le calcul de
c         la constante kappa
c
c
c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observées sur chaque individu
c     n : Nombre d'individus
c
c     alpha: représente le niveau de signification du test d'hypothèses
c
c     talpha: est le seuil du test d'hypothèses (P(T .leq. talpha)=alpha
c     sous H 0)
c
c     ncli et nlcj sont les indices des classes qui réalisent le minimum
c     de la matrice T
c
c     nom(n) : Vecteur contenant le nom associé à chaque individu (nom(i)
c              contient le nom associé au ième individu du fichier)
c
c     x(dim,n) : Matrice contenant les caractéristiques de tous les individus
c                ( x(j,i) contient la caractéristique j du ième individu )
c
c     cl(n) : Vecteur qui contient le numéro de la classe des individus
c             ( cl(i) est le numéro de la classe du ième individu )
c
c     nb(0:k) : Vecteur qui contient l'indice du dernier individu de chaque
c               classe ( nb(i) est l'indice du dernier individu de la ième
c               classe )
c
c     c(dimmax,npmax) : Matrice qui contiendra les caractéristiques des
c                       individus appartenant à la même classe
c
c     d(dimmax,npmax) : Matrice qui contiendra les caractéristiques des
c                       individus de la réunion des deux classes.
c
c     y : contient le nombre d'éléments d'une classe de la partition
c     z : contient le nombre d'éléments d'une classe de la partition
c     f : contient la valeur de nb(i-1)
c     g : contient la valeur de nb(j-1)
c
c     volume : contient l'hypervolume de la ième classe à la ième itération
```

```
c
c     vol : contient l'hypervolume des classes, vol(i) contient
c           l'hypervolume de la ième classe
c
c     MD : contient l'hypervolume de la classe formée par la réunion
c          de la classe i et la classe j
c
c     T(dimmax,pmax): matrice qui contient le calcul de la statistique T
c                     du test pour chaque paire de classes
c
c     m: représente la somme du nombre d'individus dans la classe i
c        et du nombre d'individus de la classe j
c
c     min: contient la valeur du minimum de la matrice T
c
c     kappa: contient la constante kappa nécessaire au calcul de T
c
c     i,j : indice de boucle
c
c
c     gaptest fait appel à la sous-routine volclasse (k,dim,n,z,f,x,c,volume)
c     qui calcule l'hypervolume de l'enveloppe convexe de chacune des classes.
c
c     gaptest fait aussi appel à la sous-routine
c     voldeuxclasses (k,dim,n,y,z,f,g,x,c,volume) qui calcule l'hypervolume
c     de l'enveloppe convexe de la réunion de deux classes.
c
c     gaptest utilise la sous-routine calkappa(kappa,dim,pi) qui calcule
c     la constante kappa.
c
c ***************************************************************************
c
c     Calcul de PI
c
c
      PI=4*(ATAN(1.D0))

c     appel de la sous routine qui calcule la constante kappa
c
      call calkappa (kappa,dim,pi)

c     La partition en k classes

      open (unit=20, file='colonne18.dat', status='old')
c
c     Initialisation des données
c
```

```
      do i=1,n
         do j=1,dim
            x(j,i)=0.0
            c(j,i)=0.0
            d(j,i)=0.0
         enddo
      enddo
c
c     Lecture du fichier contenant les données
c
      do  i=1,n
         read(20,*) nom (i),(x(j,i), j=1,dim),cl(i)
      enddo

      do i=1,k
         vol(i)=0.0
      enddo

c     Evaluation du vecteur nb

      j=1
      nb(0)=0
      do i=1,k
 200     continue
         if ((cl(j).eq.i).and.(j.le.n))then
            j=j+1
            goto 200
         endif
         nb(i)=j-1
      enddo

c     Calcul de l'hypervolume de l'enveloppe convexe de chaque classe.

      do i=1,k
         z=nb(i)-nb(i-1)

         f=nb(i-1)
         volume=0.0
         call volclasse (k,dim,n,z,f,x,c,volume)
         vol(i)=volume
      enddo
c
c     Calcul de l'hypervolume de l'enveloppe convexe de la réunion
c     de deux classes.


      do i=1,k
```

```
         do j=i+1,k
            y=nb(i)-nb(i-1)

c     nombre de points dans la classe i

            z=nb(j)-nb(j-1)

c     nombre de points dans la classe j

            f=nb(i-1)
            g=nb(j-1)

            MD=0.0

c     appel de la procédure qui calcule l'hypervolume de la réunion des deux
c     classes

            call voldeuxclasses (k,dim,n,y,z,f,g,x,d,MD)
            m=y+z
c
c     Calcul de la matrice T
c
            T(i,j)=((m)*(MD-vol(i)-vol(j))/MD)-(log(m))-((dim-1)*
     +           (log(log(m))))-log(kappa)

         enddo
      enddo
c
c     Calcul du minimum de la matrice T
c
      min=1000000
      do i=1,k
         do j=i+1,k
            if (T(i,j).le.min) then
               min=T(i,j)
               ncli=i
               nclj=j
            endif
         enddo
      enddo
c
c     Calcul de talpha
c
      talpha=-(log(-(log(1-alpha))))
c
      do i=1,k
         do j=i+1,k
```

```
            write(*,*) 'T(',i,j,')',T(i,j)
         enddo
      enddo

      write(*,*) 'min', min
      write(*,*) 'talhpa',talpha
c
c     Résultat du test
c
      if (k.eq.1) then
         write(*,*) 'La partition étudiée ne possède qu''une '
     + ,'seule classe donc le nombre de classes naturelles'
     + ,'présentes dans les données est 1'
      else
         if (T(ncli,nclj)< talpha) then
            write (*,*) 'on rejette la partition en ',k
     +           ,' classes'
            write(*,*) 'Il faut réappliquer le gaptest '
     +           ,'avec une partition en ',k-1,' classes.'

         else
            write (*,*) 'Le nombre de classes naturelles dans '
     +     ,'les données suivant la méthode du gaptest est ', k ,'.'
         endif
      endif
      close (20)
      end

c     ******************************************************
      SUBROUTINE volclasse (k,dim,n,z,f,x,c,volume)
c     ******************************************************
c
c     volclasse calcule l'hypervolume de l'enveloppe convexe de la ième
c     classe.

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'

c     Déclaration et description des argumemts de volclasse
c     -----------------------------------------------------


      INTEGER k,dim,n,z,f
      DOUBLE PRECISION volume
      real x(dim,n),c(dimmax,z)
```

```
c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observées sur chaque individus
c     n : Nombre d'individus
c     z : contient le nombre d'élément de la ième classe
c     f : contient la valeur de nb(i-1)
c     x(dim,n) : Matrice contenant les caractéristiques de tous les individus
c                ( x(j,i) contient la caractéristique j du ième individu
c     c(dimmax,npmax) : Matrice qui contient les caractéristiques des
c                       individus appartenant à la même classe
c     volume : contient l'hypervolume


c     Déclaration et description des variables utilisées dans volclasse
c     -----------------------------------------------------------------

      INTEGER l,j

c     l,j : indice de boucles

c     volclasse fait appel à la sous-routine convexe. La déclaration de ses
c     arguments est :

      INTEGER          face    (dimmax,nfmax)
      INTEGER hyper1 (3,nhmax)
      INTEGER face1  (nfmax)
      INTEGER nh,nf,h
      DOUBLE PRECISION surface
      DOUBLE PRECISION hyperplan (dimmax,nhmax)
      DOUBLE PRECISION vectra    (dimmax)


      do l=1,dim
         do j=1,z
            c(l,j)=x(l,f+j)
         enddo
      enddo

      call convexe (c,hyperplan,vectra,face,hyper1,face1,
     +     dim,z,nh,nf,volume,h,surface,.false.)
      end


c     ****************************************************************
      Subroutine voldeuxclasses (k,dim,n,y,z,f,g,x,c,volume)
c     ****************************************************************
c     cette sousroutine calcule la mesure de l'enveloppe convexe de la classe
```

```
c     formée par les points des deux classes Ci et Cj.

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'

c     Declaration et description des argumemts de voldeuxclasses
c     ----------------------------------------------------------


      INTEGER k,dim,n,y,z,f,g
      DOUBLE PRECISION volume
      Real x(dim,n),c(dimmax,y+z)
      integer r
c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observées sur chaque individus
c     n : Nombre d'individus
c     y : contient le nombre d'élément de la jème classe
c     z : contient le nombre d'élément de la jème classe
c     f : contient la valeur de nb(i-1)
c     g : contient la valeur de nb(j-1)
c     x(dim,n) : Matrice contenant les caractéristiques de tous les individus
c                ( x(j,i) contient la caractéristique j du ième individu
c     c(dimmax,npmax) : Matrice qui contient les caractéristiques des
c                       individus appartenant à la même classe
c     volume : contient l'hypervolume


c     Déclaration et description des variables utilisées dans volclasse
c     -----------------------------------------------------------------

      INTEGER l,j

c     l,j : indice de boucles

c     voldeuxclasses fait appel à la sous-routine convexe.
c     La déclaration de ses arguments est :
c

      INTEGER face    (dimmax,nfmax)
      INTEGER hyper1 (3,nhmax)
      INTEGER face1  (nfmax)
      INTEGER nh,nf,h
      DOUBLE PRECISION surface
      DOUBLE PRECISION hyperplan (dimmax,nhmax)
      DOUBLE PRECISION vectra    (dimmax)
```

```
c     initialisation de la matrice c avec les éléments
c     de la classe i et la classe j

      do l=1,dim
         do j=1,y
            c(l,j)=x(l,f+j)
         enddo
         do j=1,z
            c(l,y+j)=x(l,g+j)
         enddo
      enddo

      r=y+z
      call convexe (c,hyperplan,vectra,face,hyper1,face1,
     +     dim,r,nh,nf,volume,h,surface,.false.)
      end

c     ***********************************************************************
      subroutine calkappa(kappa,dim,pi)
c     ***********************************************************************
c
c     Cette sous-routine calcule la valeur de la constante kappa
c     utilisée dans le calcul de la matrice T
c
c
      implicit none

      integer dim
      double precision kappa,pi,gammanum,gammaden
      integer factor
      double precision gamma
c
c     Description des variables utilisées dans calkappa
c     -------------------------------------------------
c
c     dim: Nombre de variables observées sur chaque individus
c
c     kappa: valeur de la constante kappa déterminée après la sous-routine
c
c     pi: valeur de pi
c
c     gammanum: partie du numérateur de kappa
c
c     gammden: partie du dénominateur de kappa
c
```

```
c     La sous-routine calkappa utilise la sous-routine calgamma(dim,gamma,pi)
c     qui calcule la fonction gamma de n/2 où n est entier.
c
c     Cette sous-routine utilise les variables dim, pi et gamma qui
c     contient la valeur de la fonction gamma en n/2
c
c     La sous-routine calkappa utilise aussi la fonction factor(entier) qui
c     calcule la factorielle de l'entier
c
c     Pour calculer kappa on distingue les cas où dim est pair et dim est
c     impair
c
c
c     si dim est pair

      if ((dim/2).eq.(real(dim)/2)) then

         gammanum=factor(dim/2)

         call calgamma(dim+1,gamma,pi)
         gammaden=gamma


         kappa= (1/real(factor(dim)))*((((sqrt(pi))*(gammanum))/
     +              (gammaden))**(dim-1))
         write(*,*) 'kappa=', kappa
c
c     si dim est impair
c
      else
         gamma=0
         call calgamma(dim,gamma,pi)
         gammanum=(real(dim)/2)*gamma

         gammaden=factor(((dim+1)/2)-1)


         kappa= (1/real(factor(dim)))*((((sqrt(pi))*(gammanum))/
     +              (gammaden))**(dim-1))
         write(*,*) 'kappa=', kappa

      endif
      end
c
c
c     ***********************************************************************
      subroutine calgamma(dim,gamma,pi)
```

```
c     ***********************************************************************
c
c     Cette sous-routine permet de calculer la valeur de la fonction
c     gamma de l'entier (dim/2)
c
c
      implicit none

      integer dim
      double precision gamma
      integer doublefactor
      double precision pi
c
c
c     Description des variables utilisées dans calgamma
c     -------------------------------------------------
c
c     dim: Nombre de variables observées sur chaque individu
c
c     gamma : contient la valeur de la fonction gamma de dim/2
c
c     pi: valeur de la constante pi
c
c     La sous-routine utilise la fonction doublefactor(entier)
c     qui calcule la valeur de la double factorielle de l'entier
c
c
      gamma = (doublefactor(dim-2)*sqrt(pi))/(2**((real(dim)-1)/2))

      end
c
c
c     ***********************************************************************
      integer function factor(entier)
c     ***********************************************************************

      implicit none

      integer entier
      integer i
c
c     Description des variables de la fonction factor(entier)
c     -------------------------------------------------------
c
c     entier : l'entier pour lequel on calcule la factorielle
c
c     i: indice de boucle
```

```
c
c
      factor=1
      do i=entier,1,-1
         factor=factor*i
      enddo
      end


c     ***********************************************************************
      integer function doublefactor(entier)
c     ***********************************************************************

      implicit none

      integer entier
      integer i
c
c     Description des variables de la fonction factor(entier)
c     -------------------------------------------------------
c
c     entier : l'entier pour lequel on calcule la double factorielle
c
c     i: indice de boucle
c
c
      doublefactor=1
      do i=entier,1,-2
         doublefactor=doublefactor*i
      enddo
      end
c
c     ***********************************************************************
c     ***********************************************************************
      SUBROUTINE CONVEXE (POINT,HYPERPLAN,VECTRA,FACE,HYPER1,
     +                    FACE1,DIM,NP,NH,NF,VOLUME,H,SURFACE,COND)
C
C     ***********************************************************************
C     ***********************************************************************
C
C     BUT DE LA SOUS-ROUTINE :
C     ------------------------
C
C           Calculer l'enveloppe convexe d'un ensemble de points d'un
C espace de dimension DIM. La méthode utilisée n'est pas recursive.
C
C     ***********************************************************************
```

```
C
      IMPLICIT NONE
C
      INCLUDE 'CONSTMAX.DEF'
      INCLUDE 'FATAL_LIMITS.DEF'
C
C
C SPECIFICATION DES ARGUMENTS DE LA SOUS-ROUTINE :
C ------------------------------------------------
C
Real           POINT     (DIMMAX,*)
DOUBLE PRECISION HYPERPLAN (DIMMAX,*)
DOUBLE PRECISION VECTRA    (DIMMAX)
INTEGER          FACE      (DIMMAX,*)
C
INTEGER HYPER1 (3,*)
INTEGER FACE1  (*)
C
INTEGER          DIM
INTEGER          NP,NH,NF
INTEGER          H
DOUBLE PRECISION VOLUME
DOUBLE PRECISION SURFACE
LOGICAL          COND
C
C
C DESCRIPTION DES ARGUMENTS DE LA ROUTINE
C ---------------------------------------
C
C       DIMMAX : variable entière contenant la dimension de déclaration des
C          tableaux dans le programme appelant. Cela correspond à la
C          dimension maximale du problème.
C       NPMAX : nombre maximum de points.
C       NCBMAX : nombre maximum de clusters.
C       NHMAX : nombre maximum d'hyperplans.
C       NFMAX : nombre maximum de faces.
C POINT : tableau de dimension 2 d'entiers. Il contient les coordonnées
C         des différents points dont il faut chercher l'enveloppe
C         convexe. Les points (au nombre de NP) appartiennent à un
C               espace de dimension DIM. Ils sont donc caractérisés par DIM
C               entiers (DIM <=DIMMAX).
C HYPERPLAN : tableau de dimension 2 de reels. Il contient les
C             coefficients des équations des hyperplans.
C             A la sortie de la routine, il y aura NH hyperplans.
C             Les équations seront de la forme :
C                           somme   (alpha  * x ) = 1 ou 0
C                          i=1,dim        i    i
```

```
C            Un hyperplan, dans un espace de dimension DIM, est donc
C            caractérisé par dim coefficients 'alpha' et un terme
C            indépendant. Le terme indépendant se trouvera, à la
C            sortie, dans la première ligne du tableau HYPER1.
C            Les coefficients 'alpha' seront dans le tableau HYPERPLAN
C            Les 'x' correspondent aux coordonnées d'un point. Le terme
C            indépendant peut être 0 ou 1 suivant que l'hyperplan
C            passe ou non par l'origine (0,...,0).
C            HYPERPLAN(I,J) est le ième coeffic. de l'hyperplan
C            numéro j.
C       VECTRA : tableau de dimension 1 de réels. Il s'agit d'un vecteur de
C          vecteur de translation utilisé pour que le point de
C          coordonnées (0,0,...,0) soit intérieur au polyedre de base.
C          Ceci facilite la recherche des équations des hyperplans.
C          Si COND est 'true', les équations des hyperplans seront
C                relatives à ce vecteur. Sinon, elles seront absolues (i.e.
C                relatives à la vraie origine (0,...,0)).
C       FACE : tableau de dimension 2 d'entiers. Il contient les références
C        des points composant une face. Cette référence correspond au
C        tableau POINT. Il faut faire très attention au fait que pour
C              nous, une face est un ensemble de DIM points différents.
C              Par exemple :
C                     - en dimension 1 : face = un point
C                     - en dimension 2 : face = un segment
C                     - en dimension 3 : face = un triangle
C                     - en dimension 4 : face = un tetraedre
C                     - ...
C        De ce fait, un même hyperplan peut contenir plusieurs faces.
C              FACE(I,J) est le numéro du ième point de la jème face.
C       HYPER1 : tableau de dimension 2 d'entiers.
C          HYPER1(1,J) = statuts de l'hyperplan j.
C                      = 0 : l'hyperplan est interne au polyedre courant.
C                      = 1 : l'hyperplan est transitoire.
C                      = 2 : l'hyperplan est final.
C          HYPER1(2,J) = nbre de faces de l'hyperplan j
C  HYPER1(3,j) = no de la 1er face de l'hyperplan j
C          A la sortie de la sous-routine, seuls les hyperplans
C          finaux sont conserves. Dès lors, la première ligne de
C          HYPER1 n'a plus de raison d'être. On y met le terme in-
C          dépendant de l'équation de l'hyperplan.
C       FACE1 : tableau de dimension 1 d'entiers contenant la référence
C         d'une face appartenant au même hyperplan. S'il n'y en a pas
C         d'autre, sa valeur est 0.
C         FACE1(I) = no de la face suivante appartenant au même hyperplan
C                    que la face i
C       DIM : variable entière contenant la dimension du problème.
C NP : variable entière contenant le nombre de points dont il faut
```

```
C      chercher l'enveloppe convexe.
C NH : variable entière contenant le nombre courant d'hyperplans.
C            plans. A la sortie, cette variable contiendra le nombre
C            d'hyperplans finaux.
C       NF : variable entière contenant le nombre courant de faces. A la
C      sortie, cette variable contiendra le nombre de faces inclues dans
c        les hyperplans finaux.
C       VOLUME : variable DOUBLE PRECISION contenant les hypervolumes des
C                différents convexes.
C SURFACE: variable DOUBLE PRECISION contenant la surface du polyhèdre
C       COND : variable logique étant a 'true' si les équations des
C        hyperplans, à la sortie, sont relatives à VECTRA et
C        'false' si elles sont absolues.
C
C
C DECLARATION DES CONSTANTES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------------
C
DOUBLE PRECISION PREC1,PREC2,PREC
C
PARAMETER (PREC1=0.,PREC2=0.,PREC=1D-10)
C
C
C LISTE DES CONSTANTES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------
C
C       NHEMAX : constante entiere contenant le nombre maximal d'hyperplans
C          vis-a-vis desquelles un meme point peut etre exterieur.
C       NFEMAX : constante entiere contenant le nombre maximal de faces
C          vis-a-vis desquelles un meme point peut etre exterieur.
C NNFMAX : constante entiere contenant le nombre maximal de nouvelles
C          faces pouvant etre engendrees par un meme point.
C PREC1 : constante DOUBLE PRECISION contenant la precision desiree
C         lors du calcul des equations des hyperplans.
C PREC2 : constante DOUBLE PRECISION contenant la precision desiree
C         lors de la comparaison des equations des hyperplans.
C       LIMIT_RATIO : constante entiere contenant la proportion de points
C       entrant dans la construction de l'enveloppe convexe
C                      Cette constante peut donc etre utilisee pour obtenir
C                      une approximation de l'enveloppe convexe par
C       l'interieur. Elle n'est utilisee ici qu'a partir de
C       DIM = 6 et peut valoir par exemple 90 ou 95.
C
C
C DECLARATION DES VARIABLES UTILISEES DANS LA SOUS-ROUTINE :
C ----------------------------------------------------------
C
```

```
INTEGER NPS,NPI
INTEGER NHT,NHI,NHF
C
INTEGER NFE,     FACEXT (NFEMAX)
INTEGER NNF,     NOUFAC (DIMMAX,NNFMAX)
INTEGER NH1,NH2,LIHYIN(NHEMAX),LIHYEX(NHEMAX)
C
INTEGER INITIA (DIMMAX)
INTEGER HT,PT
C
INTEGER          X(DIMMAX),Y(DIMMAX),TERIND
INTEGER          POINT1(NPMAX)
DOUBLE PRECISION POINT2(DIMMAX,NPMAX)
DOUBLE PRECISION BASE(DIMMAX,DIMMAX)
DOUBLE PRECISION A(DIMMAX,DIMMAX),B(DIMMAX)
C
INTEGER          I,J,K,L,M
C INTEGER          I1,I2,I3,I4,I5,I6
DOUBLE PRECISION SOMME,DET(2),AUX
double precision cg(DIMMAX),vol,cgaux(DIMMAX)
LOGICAL          EGAL

INTEGER LIMIT_RATIO
PARAMETER (LIMIT_RATIO = 100)
INTEGER NP_LIMIT_RATIO
C
C
C LISTE DES VARIABLES UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------
C
C       POINT1 : tableau de dimension 1 d'entiers. Il contient les statuts
C          associes a chaque point. On considere qu'il y a trois
C          statuts possibles durant l'execution de la sous-routine
C                    . -1 : le point n'a pas encore ete traite.
C                    .  0 : le point est interne au polyedre courant.
C                    . +1 : le point est sommet du polyedre courant.
C          A la sortie de la sous-routine, seuls les deux derniers
C          statuts restent actifs (i.e. on associe a chaque point
C          un des deux derniers statuts).
C NPS : variable entiere contenant le nombre de points consideres
C       comme etant des sommets (ou pseudo-sommets).
C       NPI : variable entiere contenant le nombre de points consideres
C       comme etant internes.
C NHT : variable entiere contenant le nombre d'hyperplans, parmi
C       les NH hyperplans, consideres comme etant transitoires.
C NHI : variable entiere contenant le nombre d'hyperplans, parmi
C            les NH hyperplans, consideres comme etant internes.
```

```
C NHF : variable entiere contenant le nombre d'hyperplans, parmi
C             les NH hyperplans, consideres comme etant finaux.
C NFE : variable entiere contenant le nombre de faces pour lesquelles
C       le point PT est exterieur.
C FACEXT : tableau de dimension 1 d'entiers. Il contient les references
C          des faces pour lesquelles le point PT est exterieur.
C          Les references correspondent au tableau FACE. Ce tableau peut
C          contenir au maximum NFEMAX de references.
C       NNF : variable entiere contenant le nombre de nouvelles faces
C       formees a partir du point PT de de l'hyperplan HT.
C       NOUFAC : tableau de dimension 2 d'entiers. Il contient les references
C          des points composant les nouvelles faces formees a partir
C          du point PT et de l'hyperplan HT. Les references
C          correspondent au tableau POINT. Ce tableau peut
C          contenir au maximum NNFMAX faces, decrites par maximum
C          DIMMAX points.
C NH1 : variable entiere contenant le nombre d'hyperplans pour
C       lesquelles le point PT devra etre interieur (pour etre
C       inclus dans le convexe).
C       NH2 : variable entiere contenant le nombre d'hyperplans pour
C       lesquelles le point PT devra etre exterieur (pour etre
C       inclus dans le convexe).
C LIHYIN : tableau de dimension 1 d'entiers. Il contient les references
C          des hyperplans pour lesquels le point PT est interieur.
C LIHYEX : tableau de dimension 1 d'entiers. Il contient les references
C          des hyperplans pour lesquels le point PT est exterieur.
C INITIA : tableau de dimension 1 d'entiers. Il est utilise dans la
C          construction du polyedre de base. Il contient les references
C          des differents points composant le polyedre de base.
C          Les references correspondent au tableau POINT.
C HT : variable entiere contenant l'indice du premier hyperplan
C      transitoire non encore traite (hyperplan transitoire courant).
C PT : variable entiere contenant l'indice du point le plus exterieur
C      a l'hyperplan HT.
C X : tableau de dimension 1 d'entiers. Il est utilise dans la recherche
C     des equations des hyperplans, et plus particulierement dans la
C     resolution du systeme d'equations. Il est egalement utilise lors
C           de la recherche du volume.
C Y : tableau de dimension 1 d'entiers. Il est utilise dans
C     la recherche des equations des hyperplans, et plus
C     particulierement dans la resolution du systeme d'equations.
C TERIND : variable entiere contenant le terme independant de l'
C          equation d'un hyperplan. Ce terme peut prendre deux
C          valeurs possibles : 0 si l'hyperplan passe par l'origine
C          et 1 sinon.
C POINT2 : tableau de dimension 2 de reels. Il contient les
C          cooordonnees des differents points. Il peut donc contenir au
```

```
C                maximum NPMAX points d'un espace de dimension maxim
C BASE : tableau de dimension 2 de reels. Il est utilise pour la
C              determination du polyedre de base, plus particulierement pour
C              la recherche des differents sommets constituant le polyedre de
C              base.
C A : tableau de dimension 2 de reels. Il est utilise dans la recherche
C           des equations des hyperplans, et ce plus particulierement dans la
C           resolution du systeme d'equations.
C B : tableau de dimension 1 de reels. Il est utilise dans la recherche
C           des equations des hyperplans, et ce plus particulierement dans la
C           resolution du systeme d'equations.
C I,J,K,L,M : variables entieres utilisees comme indices de boucle.
C SOMME : variable DOUBLE PRECISION contenant le resultat de la
C         fonction SOM.
C DET : tableau de dimension 1 de reels. Il est utilise dans le calcul
C             du determinant.
C AUX : variable DOUBLE PRECISION auxilliaire.
C       EGAL : variable logique utilisee lors de la comparaison des
C        equations des hyperplans
C
C
C DECLARATION DES FONCTIONS UTILISEES DANS LA SOUS-ROUTINE :
C -----------------------------------------------------
C
DOUBLE PRECISION SOM
INTEGER          REPOEL,REPOEX
LOGICAL          INT,EXT
C
EXTERNAL  SOM,REPOEL,REPOEX,INT,EXT
INTRINSIC ABS
C
C
C LISTE DES FONCTIONS ET ROUTINES UTILISEES DANS LA SOUS-ROUTINE :
C ------------------------------------------------------------
C
C       SOM (DIM,POINT,HYPERPLAN) : fonction DOUBLE PRECISION qui
C     calcule le produit scalaire du vecteur POINT avec le
C     vecteur HYPERPLAN, les deux vecteurs etant de dimension DIM.
C       REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,TERIND) : fonction
C           entiere qui renvoie l'indice du point le plus eloigne de
C           l'hyperplan specifie. La recherche du point ne s'effectue que
C           sur les points exterieurs au polyedre courant.
C REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN) : fonction entiere
C     qui renvoie l'indice du point le plus exterieur
C     a l'hyperplan specifie. La recherche du point ne s'
C     effectue que sur les points exterieurs au polyedre courant.
C INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
```

```
C     qui indique si le point specifie est interieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est exterieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,DIM,NNF,NOUFAC) :
C     sous-routine qui recherche les nouvelles faces engendrees par le
C          point PT et par les NFE faces dont les indices se trouvent dans
C          le tableau FACEXT. Cette routine exclut les faces internes.
C REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,DIM,TERIND,PREC) :
C     sous-routine qui recherche l'equation d'un hyperplan.
C     Plus particulierement, elle recherche les coefficients de
C     l'hyperplan devant contenir les points de la face specifiee
C     Les coefficients obtenus constituent l'hyperplan
C     specifie. La variable TERIND contient la valeur du terme
C     independant de l'equation de l'hyperplan.
C       PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,HYPER1,FACE1,DIM,DIMMAX) :
C          sous-routine qui purge les hyperplans et les faces et nettoye
C          les tableaux.
C
C ********************************************************************
C *  INITIALISATION :                                                *
C *                                                                  *
C *     - initialisation de POINT2, de POINT1 et de VECTRA           *
C ********************************************************************
C BEGIN
C *****
c
        Fatal_Limit_NHMAX = NHMAX-1
        Fatal_Limit_NFMAX = NFMAX-DIMMAX
        Fatal_Limit_NPMAX = NPMAX



FATAL_LIMIT_NHEMAX = NHEMAX - 1
FATAL_LIMIT_NFEMAX = NFEMAX - 1
FATAL_LIMIT_NNFMAX = NNFMAX - 1

        IF (NP .GT. NPMAX) THEN
c     WRITE (MSG_UNIT,*) 'NP=', NP
c     WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
    STOP
ENDIF
c
```

```
C ------------------------------------------------
C initialisation de POINT2, de POINT1 et de VECTRA
C ------------------------------------------------
C
if (np .lt. 20) then
c           write (MSG_UNIT,*) ' # points:', np
c     write (MSG_UNIT,*) '**********************'
           do i=1,np
c              write (MSG_UNIT,'(3i6)') ( point (j,i) ,j=1, dim)
           enddo
        endif
C
C On construit une fenetre autour de chaque point, c.a.d qu'on ajoute
C autour de chaque point les sommets d'un cube centre en ce point, et
C dont la longueur du demi-cote vaut h
C
K=NP
c DO I=1,NP
c
c
c   IF (DIM.EQ.1) THEN
c           DO I1=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)        ! facteur h...
c     ENDDO
c   ENDIF
c   IF (DIM.EQ.2) THEN
c     DO I1=0,1
c     DO I2=0,1
c        K=K+1
c        IF (K .GT. NPMAX) THEN
c           WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c           STOP
c        ENDIF
c        POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c        POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c     ENDDO
c     ENDDO
c   ENDIF
c   IF (DIM.EQ.3) THEN
c     DO I1=0,1
c     DO I2=0,1
```

```
c           DO I3=0,1
c         K=K+1
c         IF (K .GT. NPMAX) THEN
c            WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c            STOP
c         ENDIF
c         POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c         POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c         POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c      ENDDO
c      ENDDO
c      ENDDO
c    ENDIF
c      IF (DIM.EQ.4) THEN
c      DO I1=0,1
c      DO I2=0,1
c      DO I3=0,1
c           DO I4=0,1
c         K=K+1
c         IF (K .GT. NPMAX) THEN
c            WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c            STOP
c         ENDIF
c         POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c         POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c         POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c         POINT(4,K)=POINT(4,I)+H*(2*I4-1)
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c    ENDIF
c    IF (DIM.EQ.5) THEN
c      DO I1=0,1
c      DO I2=0,1
c      DO I3=0,1
c      DO I4=0,1
c           DO I5=0,1
c         K=K+1
c         IF (K .GT. NPMAX) THEN
c            WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c            STOP
c         ENDIF
c         POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c         POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c         POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c         POINT(4,K)=POINT(4,I)+H*(2*I4-1)
```

```
c         POINT(5,K)=POINT(5,I)+H*(2*I5-1)
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c    ENDIF
c    IF (DIM.EQ.6) THEN
c      DO I1=0,1
c      DO I2=0,1
c      DO I3=0,1
c      DO I4=0,1
c      DO I5=0,1
c            DO I6=0,1
c         K=K+1
c         IF (K .GT. NPMAX) THEN
c            WRITE (MSG_UNIT,*) ' CONVEXE> NPMAX!...'
c            STOP
c         ENDIF
c         POINT(1,K)=POINT(1,I)+H*(2*I1-1)
c         POINT(2,K)=POINT(2,I)+H*(2*I2-1)
c         POINT(3,K)=POINT(3,I)+H*(2*I3-1)
c         POINT(4,K)=POINT(4,I)+H*(2*I4-1)
c         POINT(5,K)=POINT(5,I)+H*(2*I5-1)
c         POINT(6,K)=POINT(6,I)+H*(2*I6-1)
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c      ENDDO
c    ENDIF
c ENDDO
c
c
DO I=1,NP
   DO J=1,DIM
      POINT2 (J,I) = POINT (J,I)
           ENDDO
   POINT1 (I) = -1
    ENDDO



DO I=1,DIM
   VECTRA (I) = 0.
   cg (i) = 0.
```

```
    ENDDO
C
C ***********************************************************************
C *  CONSTRUCTION DU POLYEDRE DE BASE :                                 *
C *                                                                     *
C *     - recherche des DIM+1 sommets de base                           *
C *     - tri par ordre croissant des DIM+1 sommets de base             *
C *     - determination du centre de gravite et translation des points  *
C *     - determination des DIM+1 faces de base                         *
C *     - determination des DIM+1 hyperplans de base                    *
C *     - recherche des points internes                                 *
C ***********************************************************************
C
C ---------------------------------
C recherche des DIM+1 sommets de base
C ---------------------------------
C
C initialisation de BASE, de FACE(.,1) et de HYPERPLAN(.,1)
C
DO I=1,DIM
   DO J=1,DIM
      BASE (J,I) = 0.
   50    ENDDO
   BASE (I,I) = 1.
   FACE (I,1) = I
   HYPERPLAN (I,1) = 1.
   40 ENDDO
TERIND = 1
C
C recherche des DIM sommets formant l'hyperplan de base
C
EGAL = .TRUE.
I = 0
   60 CONTINUE
IF (EGAL.AND.(I.LT.DIM)) THEN
C
C    rechercher le point le plus eloigne de l'hyperplan de base
   PT = REPOEL (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,1),
     +                 TERIND)
C
c    write(MSG_UNIT,*) 'after repoel, np=',np
C    former le nouvel hyperplan de base en tenant compte du point trouve
   IF (PT.GT.0) THEN
      I = I + 1
      INITIA (I) = PT
      POINT1 (PT) = 1
      DO J=1,DIM
```

```
         BASE (J,I) = POINT2 (J,PT)
   70       ENDDO
C
C       recherche des coefficients de l'equation de l'hyperplan de base
        CALL REEQHY (BASE,FACE(1,1),HYPERPLAN(1,1),A,B,X,Y,
     +                    DIMMAX,DIM,TERIND,PREC1)
C
c       write(MSG_UNIT,*) 'after REEQHY'
C
   ELSE
      EGAL = .FALSE.
   ENDIF
   GOTO 60
ENDIF
C
C recherche du DIM+1 ieme sommet de base
C
IF (EGAL) THEN
C
C    rechercher le point le plus eloigne de l'hyperplan de base
   PT = REPOEL (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,1),
     +                 TERIND)
c    write(MSG_UNIT,*) 'after repoel, np=',np
   IF (PT.GT.0) THEN
      I = I + 1
      INITIA (I) = PT
      POINT1 (PT) = 1
   ENDIF
ENDIF
C
C s'il n'est pas possible de construire le polyedre de base
C
c write (MSG_UNIT,*) ' #pts polyedre base:',i
IF (I.LT.DIM+1) THEN
   VOLUME = 0.
   IF (I.EQ.DIM) THEN
      NH = 1
      NF = 1
      J = 0
      DO I=1,NP
         IF (POINT1(I).EQ.-1) THEN
            POINT1(I) = 0
         ELSE
            J = J + 1
            FACE(J,1) = I
         ENDIF
   80       ENDDO
```

```
      FACE1(I) = 0
      HYPER1(1,1) = TERIND
      HYPER1(2,1) = 1
      HYPER1(3,1) = 1
   ELSE
      NH = 0
      NF = 0
      DO I=1,NP
         POINT1(I) = 0
   85    ENDDO
   ENDIF
ELSE
C
   NPS = DIM + 1


c
C    ------------------------------------------------
C    tri par ordre croissant des DIM+1 sommets de base
C    ------------------------------------------------
C
   DO I=DIM,1,-1
      DO J=1,I
         IF (INITIA(J).GT.INITIA(J+1)) THEN
            K = INITIA (J)
            INITIA (J) = INITIA (J+1)
            INITIA (J+1) = K
         ENDIF
  100    ENDDO
   90  ENDDO
C
C    ------------------------------------------------------------
C    determination du centre de gravite et translation des points
C    ------------------------------------------------------------
C
C    determination du centre de gravite
C
   DO I=1,DIM+1
      K = INITIA (I)
      DO 120 J=1,DIM
         VECTRA (J) = VECTRA (J) + POINT2 (J,K)
  120    ENDDO
  110  ENDDO
C
   DO I=1,DIM
      VECTRA (I) = VECTRA (I) / (DIM+1)
      cg (i) = vectra (i)
```

```
  130    ENDDO
C
C    translation des points
C
   DO 140 I=1,NP
      DO 150 J=1,DIM
         POINT2 (J,I) = POINT2 (J,I) - VECTRA (J)
  150       CONTINUE
  140    CONTINUE
C
c    write(MSG_UNIT,*) 'VECTRA =',(vectra(j),j=1,dim)
C
C    ------------------------------------
C    calcul du volume du polyedre de base
C    ------------------------------------
C
   VOLUME = 0.
   K = INITIA (DIM+1)
   DO 160 I=1,DIM
      L = INITIA (I)
      DO 170 J=1,DIM
         A (J,I) = POINT2 (J,L) - POINT2 (J,K)
  170       CONTINUE
  160    CONTINUE
   CALL DGEFA (A,DIM,X,I)
C
c    write(MSG_UNIT,*) 'after DGEFA'
C
   IF (I.EQ.0) THEN
      CALL DGEDI (A,DIM,X,DET,B,10)
      VOLUME = VOLUME + ABS(DET(1)*10.0**DET(2))
   ENDIF


c           write(*,*)'volume=',volume
C    ------------------------------------
C    determination des DIM+1 faces de base
C    ------------------------------------
C
   NF = DIM + 1
   DO 180 I=1,NF
      FACE1 (I) = 0
      DO 190 J=1,DIM
         IF (J.LT.I) THEN
            FACE (J,I) = INITIA (J)
         ELSE
            FACE (J,I) = INITIA (J+1)
```

```
        ENDIF
  190      CONTINUE
  180   CONTINUE

C
C    ----------------------------------------
C    determination des DIM+1 hyperplans de base
C    ----------------------------------------
C
     NH  = NF
     NHT = NH
     NHI = 0
     NHF = 0
     DO 200 I=1,NH
        HYPER1 (1,I) = 1
        HYPER1 (2,I) = 1
        HYPER1 (3,I) = I
        CALL REEQHY (POINT2,FACE(1,I),HYPERPLAN(1,I),A,B,X,Y,
     +                    DIMMAX,DIM,TERIND,PREC1)
C
C       write(MSG_UNIT,*) 'after REEQHY'
C
  200   CONTINUE
C
C    ----------------------------
C    recherche des points internes
C    ----------------------------
C
     DO 210 I=1,NH
        LIHYIN (I) = I
  210   CONTINUE
C
     NPI = 0
     DO 220 I=1,NP
        IF (POINT1(I).EQ.-1) THEN
           IF (INT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,NH,
     +                    LIHYIN,PREC)) THEN
              POINT1 (I) = 0
              NPI = NPI + 1
           ENDIF
        ENDIF
  220   CONTINUE
```

```
C
C ***********************************************************************
C *  CONSTRUCTION DE L'ENVELOPPE CONVEXE :                              *
C *                                                                     *
C *     - tant qu'il y a des hyperplans transitoires et des points      *
C *       a classer faire:      *
C *        - recherche du premier hyperplan transitoire : HT            *
C *        - recherche du point exterieur a l'hyperplan HT : PT         *
C *        - s'il n'existe pas de point exterieur :                     *
C *           - considerer l'hyperplan HT comme etant final             *
C *        - s'il existe un point exterieur :                           *
C *           - considerer le point PT comme etant un sommet            *
C *           - rechercher les hyperplans pour lesquels PT est exte-    *
C *             rieur, les considerer comme etant internes. Rechercher  *
C *             les faces pour lesquelles PT est exterieur              *
C *           - recherche des nouvelles faces                           *
C *           - ajouter les nouvelles faces aux faces existantes ainsi  *
C *             que leur hyperplan (s'il n'existe pas encore) ou en le  *
C *             mettant a jour (s'il existe deja)                       *
C *           - recherche des nouveaux points internes                  *
C ***********************************************************************
C
   HT = 1
   IF ( DIM .GE. 7) THEN
IF (NP .LE. 10) THEN
NP_LIMIT_RATIO = NP
ELSE
NP_LIMIT_RATIO = ( NP * LIMIT_RATIO ) / 100
ENDIF
   ELSE
NP_LIMIT_RATIO = NP
   ENDIF
c    write (MSG_UNIT,'(a,7i5)') ' CONVEXE> npi, nps, np, nhi,
c     +          nhf, nht, nh',npi,nps,np,nhi,nhf,nht,nh
   DO WHILE ((NHT.NE.0).AND.(NPS+NPI.LT.NP_LIMIT_RATIO))
C
C       ---------------------------------
C       purger si besoin est les tableaux
C       ---------------------------------
C
      IF (NHI.GE.500) THEN
c  WRITE (MSG_UNIT,*) ' CONVEXE> calling purge (NHI=500)
c     +                with param:'
C  WRITE (MSG_UNIT,*) '         NH,NHT,NHF,NF:',
C     +                              NH, NHT, NHF, NF
```

```
C
        CALL PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,
     +                      HYPER1,FACE1,DIM,DIMMAX)
C
CMHP  Purge problem
CMHP  -------------------
 IF (NH .EQ. 0) THEN
    VOLUME = 0
    RETURN
 ENDIF
CMHP  -------------------
         HT = 1
         NHI = 0
      ENDIF
C
C       ----------------------------------------
C       recherche du premier hyperplan transitoire
C       ----------------------------------------
C

      DO WHILE (HYPER1(1,HT).NE.1)
         HT = HT + 1
      ENDDO

C       ----------------------------------------------
C       recherche du point exterieur a l'hyperplan HT
C       ----------------------------------------------
C
      PT = REPOEX (NP,DIM,DIMMAX,POINT2,POINT1,HYPERPLAN(1,HT))
C
c       write(MSG_UNIT,*) 'after REPOEX'
C
C       ------------------------------------
C       s'il n'existe pas de point exterieur
C       ------------------------------------
C
      IF (PT.EQ.0) THEN
C
C          ------------------------------------------
C          considerer l'hyperplan HT comme etant final
C          ------------------------------------------
C
         HYPER1 (1,HT) = 2
         NHT = NHT - 1
         NHF = NHF + 1
C
```

```
c                write(*,*)'hyper1(1,',ht,')=',hyper1(1,ht)
C       ----------------------------
C       s'il existe un point exterieur
C       ----------------------------
C
      ELSE
C
C          ----------------------------------------
C          considerer le point PT comme etant un sommet
C          ----------------------------------------
C
c                write(*,*)'pt=',pt

          POINT1 (PT) = 1
          NPS = NPS + 1

c                write(*,*)'nps=',nps
C
C          ------------------------------------------------------------
C          recherche des hyperplans pour lesquels PT est exterieur, les
C             considerer comme etant internes
C          recherche des faces ...
C          ------------------------------------------------------------
C
          NFE = 0
          NH1 = 0
          NH2 = 0
C
C          pour chaque hyperplan
C
          DO I=1,NH
C
C             si l'hyperplan I est transitoire
C
             IF (HYPER1(1,I).EQ.1) THEN
                SOMME = SOM (DIM,POINT2(1,PT),HYPERPLAN(1,I))
C
C                si le point PT est exterieur a l'hyperplan I
C
CMH                IF (SOMME.GT.1.) THEN
                IF (SOMME.GT.(1.+PREC)) THEN
C
C                   considerer l'hyperplan I comme etant interne
C
                   HYPER1 (1,I) = 0
                   NHT = NHT - 1
                   NHI = NHI + 1
```

```
C
C                    ajouter l'hyperplan I aux hyperplans exterieurs
C                    pour PT
C
                   NH2 = NH2 + 1
  IF ( NH2 .GE. FATAL_LIMIT_NHEMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NHEMAX!...'
STOP
  ENDIF
                   LIHYEX (NH2) = I
C                    ajouter les faces aux faces exterieures pour PT
C
                   J = HYPER1 (3,I)
                   DO WHILE (J.NE.0)
                      NFE = NFE + 1
     IF ( NFE .GE. FATAL_LIMIT_NFEMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NFEMAX!...'
STOP
     ENDIF
                      FACEXT (NFE) = J
                      J = FACE1 (J)
                   ENDDO
                ENDIF
             ENDIF
             ENDDO
C
C            ------------------------------------------
C            calcul du volume ajoute au polyedre courant
C            ------------------------------------------
C
          DO 270 I=1,NFE
             L = FACEXT (I)
             DO 275 k=1,DIM
                cgaux (k) = point (k,pt)
  275           CONTINUE
             DO 280 J=1,DIM
                M = FACE (J,L)
                DO 290 K=1,DIM
                   A (K,J) = POINT2 (K,M) - POINT2 (K,PT)
                   cgaux (k) = cgaux (k) + point (k,m)
  290              CONTINUE
  280           CONTINUE
             DO 285 J=1,DIM
                cgaux (j) = cgaux (j)/(dim+1)
  285           CONTINUE
             CALL DGEFA (A,DIM,X,J)
             IF (J.EQ.0) THEN
```

```
              CALL DGEDI (A,DIM,X,DET,B,10)
C
c        write(MSG_UNIT,*) 'after DGEDI'
C
              vol = ABS(DET(1)*10.0**DET(2))
              DO 287 k=1,DIM
                 cg(k)=((cg(k)*volume)+(cgaux(k)*vol))/
     +          (volume+vol)
  287               CONTINUE
              VOLUME = VOLUME + vol
            ENDIF
  270          CONTINUE
C
C           ----------------------------
C           recherche des nouvelles faces
C           ----------------------------
C
         CALL RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,
     +                        DIM,NNF,NOUFAC)
C
C  write(MSG_UNIT,*) 'after renofa'
C
C           ------------------------------------------------------------
C           ajouter les nouvelles faces aux faces existantes ainsi que
C           leur hyperplan (s'il n'existe pas encore) ou en le mettant a
C           jour (s'il existe deja)
C           ------------------------------------------------------------
 FATAL_CURRENT_NF = FATAL_CURRENT_NF + NNF

c                write(*,*)'FATAL_CURRENT_NF=',FATAL_CURRENT_NF
c                write(*,*)'nnf=',nnf
c                WRITE(*,*)'FATAL_LIMIT_NFMAX=',FATAL_LIMIT_NFMAX

 IF ( FATAL_CURRENT_NF .GE. FATAL_LIMIT_NFMAX) THEN
    WRITE (MSG_UNIT,*) ' CONVEXE> NFMAX!...'
    STOP
 ENDIF
         DO 300 I=1,NNF
            NF = NF + 1
            DO 310 J=1,DIM
               FACE (J,NF) = NOUFAC (J,I)
  310             CONTINUE
C
C             calcul de l'equation de l'hyperplan contenant la nouvelle
C             face
C
            CALL REEQHY (POINT2,FACE(1,NF),HYPERPLAN(1,NH+1),
```

```
     +                          A,B,X,Y,DIMMAX,DIM,TERIND,PREC1)
C
C
C             voir si l'hyperplan existe deja ou non
C
              EGAL = .FALSE.
              J = NH
  320               CONTINUE
              IF ((.NOT.EGAL).AND.(J.GT.0)) THEN
                 K = 0
  330                  CONTINUE
                    K = K + 1
                    EGAL = ABS(HYPERPLAN(K,J)-HYPERPLAN(K,NH+1))
     +                           .LE.PREC2
                 IF (EGAL.AND.(K.LT.DIM)) GOTO 330
                 J = J - 1
                 GOTO 320
              ENDIF
C
C             il y aura un hyperplan de plus, interieur pour PT
C
              NH1 = NH1 + 1
IF ( NH1 .GE. FATAL_LIMIT_NHEMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NHEMAX!...'
STOP
ENDIF
C
C             si l'hyperplan existe deja
C
              IF (EGAL) THEN
                 J = J + 1
                 FACE1 (NF) = HYPER1 (3,J)
                 HYPER1 (2,J) = HYPER1 (2,J) + 1
                 HYPER1 (3,J) = NF
                 LIHYIN (NH1) = J
C
C             si l'hyperplan n'existe pas encore
C
              ELSE
                 NH = NH + 1
        FATAL_CURRENT_NH = FATAL_CURRENT_NH + 1
        IF ( FATAL_CURRENT_NH .GE. FATAL_LIMIT_NHMAX) THEN
   WRITE (MSG_UNIT,*) ' CONVEXE> NHMAX!...'
   STOP
        ENDIF
                 NHT = NHT + 1
                 FACE1 (NF) = 0
```

```
               HYPER1 (1,NH) = 1
               HYPER1 (2,NH) = 1
               HYPER1 (3,NH) = NF
               LIHYIN (NH1) = NH
            ENDIF
  300       CONTINUE
C
C           ------------------------------------
C           recherche des nouveaux points internes
C           ------------------------------------
C
C
c  write(MSG_UNIT,*) 'after recherche points internes'
C
         DO I=1,NP
            IF (POINT1(I).EQ.-1) THEN
               IF (EXT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,NH2,
     +                          LIHYEX)) THEN
                  IF (INT (POINT2(1,I),HYPERPLAN,DIMMAX,DIM,
     +                             NH1,LIHYIN,PREC)) THEN
                     POINT1 (I) = 0
                     NPI = NPI + 1
                  ENDIF
               ENDIF
            ENDIF
            ENDDO
      ENDIF
   ENDDO
C
C ***************************************************************************
C *  CLOTURE :                                                              *
C *                                                                         *
C *     - considerer les hyperplans transitoires comme finaux               *
C *     - purge des hyperplans, des faces et nettoyage des tableaux         *
C *     - translation des points                                            *
C *     - adaptation des equations des hyperplans                           *
C ***************************************************************************
C
C    ------------------------------------------------
C    considerer les hyperplans transitoires comme finaux
C    ------------------------------------------------
C
   DO  I=1,NH
      IF (HYPER1(1,I).EQ.1) THEN
         HYPER1 (1,I) = 2
      ENDIF
      ENDDO
```

```
      NHF = NHF + NHT
            NHT = 0
C
C     ------------------------------------------------------
C     calcul de la surface du convexe
C     ------------------------------------------------------
C
      SURFACE=0
      DO I=1,NH
        IF (HYPER1(1,I).EQ.2) THEN
          SOMME=0
          DO K=1,DIM
            SOMME = SOMME + HYPERPLAN(K,I)*HYPERPLAN(K,I)
          ENDDO
          SOMME = SQRT(SOMME)
          DO K=2,DIM-1
            SOMME=SOMME/K
          ENDDO
          J=HYPER1(3,I)
          DO WHILE (J.NE.0)
C
C           calcul de la surface de la face J
C
            DO K=1,DIM
              DO L=1,DIM
                A(K,L) = POINT2(L,FACE(K,J))
              ENDDO
 ENDDO

C           --- calcul du determinant de A

            AUX=0
              CALL DGEFA (A,DIM,X,K)
C
c     write(MSG_UNIT,*) 'after DGEFA'
C
            IF (K.EQ.0) THEN
              CALL DGEDI(A,DIM,X,DET,B,10)
              AUX=ABS(DET(1)*10.0**DET(2))
            ELSE
              WRITE(MSG_UNIT,*) 'CONVEXE> WARNING : face de surface
       +                          nulle'
            ENDIF

            SURFACE = SURFACE + AUX*SOMME
            J=FACE1(J)
          ENDDO
```

```
      ENDIF
    ENDDO
c           write(*,*)'surface=', surface



C     ------------------------------------------------------
C     purge des hyperplans, des faces et nettoyage des tableaux
C     ------------------------------------------------------
C
c     WRITE (MSG_UNIT,*) ' CONVEXE> calling purge (end) with param:'
c     WRITE (MSG_UNIT,*) '          NH,NHT,NHF,NF:',NH,NHT,NHF,NF
    CALL PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,HYPER1,
      +           FACE1,DIM,DIMMAX)
C
C     write(MSG_UNIT,*) 'after purge'
C
CMHP     Purge problem
CMHP     -------------------
    IF (NH .EQ. 0) THEN
VOLUME = 0
RETURN
    ENDIF
CMHP     -------------------
C
C     ---------------------------------------
C     adaptation des equations des hyperplans
C     ---------------------------------------
C
    IF (.NOT.COND) THEN
       DO I=1,NH
          AUX = 1.
          DO J=1,DIM
             AUX = AUX + HYPERPLAN (J,I) * VECTRA (J)
             ENDDO
          IF (ABS(AUX).GT.1E-6) THEN
             DO J=1,DIM
                HYPERPLAN (J,I) = HYPERPLAN (J,I) / AUX
                ENDDO
             HYPER1 (1,I) = 1
          ELSE
             HYPER1 (1,I) = 0
          ENDIF
          ENDDO
       DO I=1,DIM
          VECTRA (I) = 0.
          ENDDO
```

```
   ELSE
      DO I=1,NH
         AUX = 1.
         DO J=1,DIM
            AUX = AUX + HYPERPLAN (J,I) * (VECTRA(J)-cg(j))
            ENDDO
         IF (ABS(AUX).GT.1E-6) THEN
            DO J=1,DIM
               HYPERPLAN (J,I) = HYPERPLAN (J,I) / AUX
               ENDDO
            HYPER1 (1,I) = 1
         ELSE
            HYPER1 (1,I) = 0
         ENDIF
         ENDDO
      DO I=1,DIM
         VECTRA (I) = cg (i)
         ENDDO
   ENDIF
C
C    --------------------
C    ajustement du volume
C    --------------------
C
   AUX = 1.
   DO I=2,DIM
     AUX=AUX*I
   ENDDO
   VOLUME = VOLUME / AUX
C
ENDIF
C


c        WRITE (MSG_UNIT,*) ' CONVEXE> # hyperplans, volume', NH, VOLUME
c write (MSG_UNIT,*) 'CG=',(cg(j),j=1,dim)
c
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
DOUBLE PRECISION FUNCTION SOM (DIM,POINT,HYPERPLAN)
C
C       SOM (DIM,POINT,HYPERPLAN) : fonction DOUBLE PRECISION qui
C     calcule le produit scalaire du vecteur POINT avec le
```

```
C     vecteur HYPERPLAN, les deux vecteurs etant de dimension DIM.
C
C
IMPLICIT NONE
C
INTEGER          DIM
DOUBLE PRECISION POINT(*),HYPERPLAN(*)
C
INTEGER          I
DOUBLE PRECISION SOMME
C
SOMME = 0.
DO 10 I=1,DIM
   SOMME = SOMME + POINT(I) * HYPERPLAN(I)
   10 CONTINUE
SOM = SOMME
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
INTEGER FUNCTION REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,
     +    TERIND)
C
C
C       REPOEL (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN,TERIND) : fonction
C           entiere qui renvoie l'indice du point le plus eloigne de
C           l'hyperplan specifie. La recherche du point ne s'effectue que
C           sur les points exterieurs au polyedre courant.
C
IMPLICIT NONE
C
INTEGER          NP,DIM,DIMMAX,POINT1(*),TERIND
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*)
C
INTEGER          I,PT
DOUBLE PRECISION SOMME,DISTANCE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
INTRINSIC ABS
C
PT = 0
DISTANCE = 0.
DO 10 I=1,NP
```

```
      IF (POINT1(I).EQ.-1) THEN
         SOMME = SOM (DIM,POINT(1,I),HYPERPLAN)
         IF (ABS(SOMME-TERIND).GT.(DISTANCE+1.D-10)) THEN
            PT = I
            DISTANCE = ABS(SOMME-TERIND)
         ENDIF
      ENDIF
      10 CONTINUE
REPOEL = PT
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
INTEGER FUNCTION REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN)
C
C
C REPOEX (NP,DIM,DIMMAX,POINT,POINT1,HYPERPLAN) : fonction entiere
C     qui renvoie l'indice du point le plus exterieur
C     a l'hyperplan specifie. La recherche du point ne s'
C     effectue que sur les points exterieurs au polyedre courant.
C
IMPLICIT NONE
C
INTEGER          NP,DIM,DIMMAX,POINT1(*)
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*)
C
INTEGER          I,PT
DOUBLE PRECISION SOMME,DISTANCE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
PT = 0
DISTANCE = 0.
DO 10 I=1,NP
   IF (POINT1(I).EQ.-1) THEN
      SOMME = SOM (DIM,POINT(1,I),HYPERPLAN)
      IF ((SOMME-1.).GT.DISTANCE) THEN
         PT = I
         DISTANCE = SOMME - 1.
      ENDIF
   ENDIF
   10 CONTINUE
REPOEX = PT
```

```
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
LOGICAL FUNCTION INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP,
     +    PREC)
C
C
C INT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est interieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C
C
IMPLICIT NONE
C
INTEGER          DIMMAX,DIM,NH,LISHYP(*)
DOUBLE PRECISION POINT(*),HYPERPLAN(DIMMAX,*),PREC
C
INTEGER          I
DOUBLE PRECISION SOMME
LOGICAL          VALABLE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
VALABLE = .TRUE.
C
I = 1
   10 CONTINUE
IF (VALABLE.AND.(I.LE.NH)) THEN
   SOMME = SOM (DIM,POINT,HYPERPLAN(1,LISHYP(I)))
   VALABLE = SOMME.LE.(1.+PREC)
   I = I + 1
   GOTO 10
ENDIF
C
INT = VALABLE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
```

```
LOGICAL FUNCTION EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP)
C
C
C EXT (POINT,HYPERPLAN,DIMMAX,DIM,NH,LISHYP) : fonction logique
C     qui indique si le point specifie est exterieur
C     aux NH hyperplans referencies dans le tableau LISHYP.
C
C
IMPLICIT NONE
C
INTEGER          DIMMAX,DIM,NH,LISHYP(*)
DOUBLE PRECISION POINT(*),HYPERPLAN(DIMMAX,*)
C
INTEGER          I
DOUBLE PRECISION SOMME
LOGICAL          VALABLE
C
DOUBLE PRECISION SOM
EXTERNAL SOM
C
VALABLE = .FALSE.
C
I = 1
   10 CONTINUE
IF ((.NOT.VALABLE).AND.(I.LE.NH)) THEN
   SOMME = SOM (DIM,POINT,HYPERPLAN(1,LISHYP(I)))
   VALABLE = SOMME.GT.1.
   I = I + 1
   GOTO 10
ENDIF
C
EXT = VALABLE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE RENOFA (MSG_UNIT,PT,FACEXT,NFE,FACE,DIMMAX,
     +                     DIM,NNF,NOUFAC)
C
C
C RENOFA (MSG_UNIT, PT,FACEXT,NFE,FACE,DIMMAX,DIM,NNF,NOUFAC) :
C     sous-routine qui recherche les nouvelles faces engendrees par le
C           point PT et par les NFE faces dont les indices se trouvent dans
C           le tableau FACEXT. Cette routine exclut les faces internes.
```

```
C
C
IMPLICIT NONE

INCLUDE 'FATAL_LIMITS.DEF'
C
INTEGER PT,DIMMAX,FACEXT(*),NFE,FACE(DIMMAX,*),DIM,NNF
INTEGER NOUFAC(DIMMAX,*),MSG_UNIT
C
INTEGER I,J,K,L,M,IND
LOGICAL EXISTE
C
NNF = 0
DO 10 I=1,NFE
   IND = FACEXT(I)
   DO 20 J=1,DIM
      DO 30 K=1,DIM
         NOUFAC (K,NNF+1) = FACE (K,IND)
30       CONTINUE
      NOUFAC (J,NNF+1) = PT
C
      DO 40 K=DIM-1,1,-1
         DO 50 L=1,K
            IF (NOUFAC(L,NNF+1).GT.NOUFAC(L+1,NNF+1)) THEN
               M = NOUFAC(L,NNF+1)
               NOUFAC(L,NNF+1) = NOUFAC(L+1,NNF+1)
               NOUFAC(L+1,NNF+1) = M
            ENDIF
50             CONTINUE
40       CONTINUE
C
C       verifier si nouvelle face existe deja ou non. Si oui, il s'agit
C       d'une face interne, il faut oublier cette face ainsi que sa
C       deuxieme occurence.
C
      EXISTE = .FALSE.
      K = 0
60       CONTINUE
      IF ((.NOT.EXISTE).AND.(K.LT.NNF)) THEN
         K = K + 1
         L = 0
70          CONTINUE
            L = L + 1
            EXISTE = NOUFAC(L,NNF+1).EQ.NOUFAC(L,K)
         IF (EXISTE.AND.(L.LT.DIM)) GOTO 70
         GOTO 60
      ENDIF
```

```
      IF (EXISTE) THEN
         DO 80 L=K+1,NNF
            DO 90 M=1,DIM
               NOUFAC (M,L-1) = NOUFAC (M,L)
   90          CONTINUE
   80       CONTINUE
         NNF = NNF - 1
      ELSE
         NNF = NNF + 1
 IF ( NNF .GE. FATAL_LIMIT_NNFMAX) THEN
WRITE (MSG_UNIT,*) ' CONVEXE> NNFMAX!...'
STOP
 ENDIF
      ENDIF
C
   20    CONTINUE
   10 CONTINUE
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,
     +    DIM,TERIND,PREC)
C
C
C REEQHY (POINT,FACE,HYPERPLAN,A,B,X,Y,DIMMAX,DIM,TERIND,PREC) :
C     sous-routine qui recherche l'equation d'un hyperplan.
C     Plus particulierement, elle recherche les coefficients de
C     l'hyperplan devant contenir les points de la face specifiee
C     Les coefficients obtenus constituent l'hyperplan
C     specifie. La variable TERIND contient la valeur du terme
C     independant de l'equation de l'hyperplan.
C
C
IMPLICIT NONE
C
INTEGER          FACE(*),X(*),Y(*),DIMMAX,DIM,TERIND
DOUBLE PRECISION POINT(DIMMAX,*),HYPERPLAN(*),A(DIMMAX,*)
        DOUBLE PRECISION B(*),PREC
C
INTEGER          I,J,K,L,PT,NCSP
DOUBLE PRECISION AUX
C
INTRINSIC ABS
```

```
C
C initialisation des vecteurs A, B, X et Y
C
DO 10 I=1,DIM
   B (I) = 1.
   PT = FACE (I)
   DO 20 J=1,DIM
      A (J,I) = POINT (J,PT)
   20    CONTINUE
   X (I) = 0
   Y (I) = 0
   10 CONTINUE
TERIND = 1
C
NCSP = 0
DO 30 J=1,DIM
C
C    recherche du pivot de la Jieme colonne
C
   I = 0
   AUX = 0.
   DO 40 K=1,DIM
      IF (X(K).EQ.0) THEN
         IF ((A(J,K).NE.0.).AND.(ABS(A(J,K)).GT.AUX)) THEN
            AUX = ABS(A(J,K))
            I = K
         ENDIF
      ENDIF
   40    CONTINUE
C
   IF (I.NE.0) THEN
      X (I) = J
C
C       diviser la ligne du pivot par celui-ci
C
      AUX = A (J,I)
      DO 50 K=1,DIM
         A (K,I) = A (K,I) / AUX
   50       CONTINUE
      B (I) = B (I) / AUX
C
C       pivotage
C
      DO 60 K=1,I-1
         AUX = A (J,K)
         DO 70 L=1,DIM
            A (L,K) = A (L,K) - AUX * A (L,I)
```

```
   70          CONTINUE
            B (K) = B (K) - AUX * B (I)
   60       CONTINUE
         DO 80 K=I+1,DIM
            AUX = A (J,K)
            DO 90 L=1,DIM
               A (L,K) = A (L,K) - AUX * A (L,I)
   90          CONTINUE
            B (K) = B (K) - AUX * B (I)
   80       CONTINUE
   ELSE
      NCSP = NCSP + 1
      Y (NCSP) = J
   ENDIF
   30 CONTINUE
C
C voir si on a une matrice singuliere ou non
C
K = 1
IF (NCSP.NE.0) THEN
   TERIND = 0
   DO 100 I=1,DIM
      B (I) = 0.
      IF (X(I).NE.0) THEN
         DO 110 J=1,NCSP
            B (I) = B (I) - A (Y(J),I)
  110          CONTINUE
      ELSE
         B (I) = 1.
         X (I) = Y (K)
         K = K + 1
      ENDIF
  100    CONTINUE
ENDIF
C
C reorganisation du vecteur B et des coefficients solutions
C
DO 120 I=1,DIM
   A (X(I),1) = B (I)
  120 CONTINUE
DO 130 I=1,DIM
   IF (ABS(A(I,1)).LT.PREC) THEN
      HYPERPLAN (I) = 0.
   ELSE
      HYPERPLAN (I) = A (I,1)
   ENDIF
  130 CONTINUE
```

```
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C
SUBROUTINE PURGE (MSG_UNIT,NH,NHT,NHF,NF,HYPERPLAN,FACE,
     +                  HYPER1,FACE1,DIM,DIMMAX)
C
C       sous-routine qui purge les hyperplans et les faces, i.e. stocke les
C       hyperplans transitoires et finaux, ainsi que les faces qui leur
C       correspondent; et nettoyage des tableaux.
C
C
IMPLICIT NONE
C

INTEGER NH,NHT,NHF,NF,DIM,DIMMAX,FACE(DIMMAX,*),HYPER1(3,*)
INTEGER FACE1(*),MSG_UNIT
DOUBLE PRECISION HYPERPLAN(DIMMAX,*)
C
INTEGER I,J,K,L,M
C
C purge des hyperplans
C
K = 0
DO 10 I=1,NH
   IF ((HYPER1(1,I).EQ.1).OR.(HYPER1(1,I).EQ.2)) THEN
      K = K + 1
      DO 20 J=1,DIM
         HYPERPLAN (J,K) = HYPERPLAN (J,I)
   20       CONTINUE
      DO 30 J=1,3
         L = HYPER1 (J,I)
         HYPER1 (J,I) = HYPER1 (J,K)
         HYPER1 (J,K) = L
   30       CONTINUE
   ENDIF
   10 CONTINUE
C
c WRITE (MSG_UNIT,*) ' PURGE> # hyp finaux:', K
C
C
C purge des faces
C
DO 40 I=NHF+NHT+1,NH
```

```
      J = HYPER1 (3,I)
      50    CONTINUE
         K = FACE1 (J)
         FACE1 (J) = -1
         J = K
      IF (J.NE.0) GOTO 50
      40 CONTINUE
C
K = 1
DO 60 I=1,NHF+NHT
c           write(*,*)'i=',i
      J = HYPER1 (3,I)
c           write(*,*)'j=',j

      HYPER1 (3,I) = K
      70    CONTINUE
         IF (K.NE.J) THEN
            IF (FACE1(K).EQ.-1) THEN
               DO 80 L=1,DIM
                  FACE (L,K) = FACE (L,J)
      80             CONTINUE
               FACE1 (K) = FACE1 (J)
               FACE1 (J) = -1
            ELSE
               L = I + 1
      90             CONTINUE
               IF (L.LT.NHF+NHT) THEN
                  IF (HYPER1(3,L).NE.K) THEN
                     L = L + 1
                  ELSE
                     HYPER1 (3,L) = J
                     L = NH + 1
                  ENDIF
                  GOTO 90
               ENDIF
               IF (L.EQ.NHF+NHT) THEN
                  L = K + 1
     100             CONTINUE
                  IF (L.LT.NF) THEN
                     IF (FACE1(L).NE.K) THEN
                        L = L + 1
                     ELSE
                        FACE1 (L) = J
                        L = NF + 1
                     ENDIF
                     GOTO 100
                  ENDIF
```

```
            ENDIF
c                   write(*,*)'j et k=',j,k

CMHP     Purge problem
CMHP     ------------------------------------------------
C     WRITE (MSG_UNIT,*) ' PURGE> permute faces - J,K:', J, K
    IF (J .LT. K) THEN
WRITE (MSG_UNIT,*) ' PURGE> # Hyper forced to 0'
NH  = 0
NHF = 0
NHT = 0
NF  = 0
RETURN
    ENDIF
CMHP     ------------------------------------------------
            DO 110 L=1,DIM
               M = FACE (L,J)
               FACE (L,J) = FACE (L,K)
               FACE (L,K) = M
  110             CONTINUE
            L = FACE1 (J)
            FACE1 (J) = FACE1 (K)
            FACE1 (K) = L
         ENDIF
      ENDIF
      J = FACE1 (K)
      K = K + 1
   IF (J.NE.0) THEN
      FACE1 (K-1) = K
      GOTO 70
   ENDIF
   60 CONTINUE
C
C nettoyage des tableaux
C
DO 120 I=NHF+NHT+1,NH
   DO 130 J=1,DIMMAX
      HYPERPLAN (J,I) = 0.
  130    CONTINUE
   DO 140 J=1,3
      HYPER1 (J,I) = 0
  140    CONTINUE
  120 CONTINUE
NH = NHF+NHT
C
DO 150 I=K,NF
   DO 160 J=1,DIMMAX
```

```
      FACE (J,I) = 0
  160    CONTINUE
   FACE1 (I) = 0
  150 CONTINUE
NF = K - 1
C
END
C
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C ---------------------------------------------------------------------
C

      SUBROUTINE DGEDI(A,N,IPVT,DET,WORK,JOB)
C
      INCLUDE 'CONSTMAX.DEF'
C
      INTEGER N,IPVT(1),JOB
      DOUBLE PRECISION A(DIMMAX,1),DET(2),WORK(1)
C
C     DGEDI COMPUTES THE DETERMINANT AND INVERSE OF A MATRIX
C     USING THE FACTORS COMPUTED BY DGECO OR DGEFA.
C
C     ON ENTRY
C
C        A       DOUBLE PRECISION(DIMAX,N)
C                THE OUTPUT FROM DGECO OR DGEFA.
C
C        DIMMAX  INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C        IPVT    INTEGER(N)
C                THE PIVOT VECTOR FROM DGECO OR DGEFA.
C
C        WORK    DOUBLE PRECISION(N)
C                WORK VECTOR.  CONTENTS DESTROYED.
C
C        JOB     INTEGER
C                = 11   BOTH DETERMINANT AND INVERSE.
C                = 01   INVERSE ONLY.
C                = 10   DETERMINANT ONLY.
C
C     ON RETURN
C
```

```
C        A       INVERSE OF ORIGINAL MATRIX IF REQUESTED.
C                OTHERWISE UNCHANGED.
C
C        DET     DOUBLE PRECISION(2)
C                DETERMINANT OF ORIGINAL MATRIX IF REQUESTED.
C                OTHERWISE NOT REFERENCED.
C                DETERMINANT = DET(1) * 10.0**DET(2)
C                WITH  1.0 .LE. DABS(DET(1)) .LT. 10.0
C                OR  DET(1) .EQ. 0.0 .
C
C     ERROR CONDITION
C
C        A DIVISION BY ZERO WILL OCCUR IF THE INPUT FACTOR CONTAINS
C        A ZERO ON THE DIAGONAL AND THE INVERSE IS REQUESTED.
C        IT WILL NOT OCCUR IF THE SUBROUTINES ARE CALLED CORRECTLY
C        AND IF DGECO HAS SET RCOND .GT. 0.0 OR DGEFA HAS SET
C        INFO .EQ. 0 .
C
C     LINPACK. THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS DAXPY,DSCAL,DSWAP
C     FORTRAN DABS,MOD
C
C     INTERNAL VARIABLES
C
      DOUBLE PRECISION T
      DOUBLE PRECISION TEN
      INTEGER I,J,K,KB,KP1,L,NM1
C
C
C     COMPUTE DETERMINANT
C
      IF (JOB/10 .EQ. 0) GO TO 70
         DET(1) = 1.0D0
         DET(2) = 0.0D0
         TEN = 10.0D0
         DO 50 I = 1, N
            IF (IPVT(I) .NE. I) DET(1) = -DET(1)
            DET(1) = A(I,I)*DET(1)
C           ...EXIT
            IF (DET(1) .EQ. 0.0D0) GO TO 60
   10       IF (DABS(DET(1)) .GE. 1.0D0) GO TO 20
               DET(1) = TEN*DET(1)
               DET(2) = DET(2) - 1.0D0
```

```
            GO TO 10
   20       CONTINUE
   30       IF (DABS(DET(1)) .LT. TEN) GO TO 40
               DET(1) = DET(1)/TEN
               DET(2) = DET(2) + 1.0D0
            GO TO 30
   40       CONTINUE
   50    CONTINUE
   60    CONTINUE
   70 CONTINUE
C
C     COMPUTE INVERSE(U)
C
      IF (MOD(JOB,10) .EQ. 0) GO TO 150
         DO 100 K = 1, N
            A(K,K) = 1.0D0/A(K,K)
            T = -A(K,K)
            CALL DSCAL(K-1,T,A(1,K),1)
            KP1 = K + 1
            IF (N .LT. KP1) GO TO 90
            DO 80 J = KP1, N
               T = A(K,J)
               A(K,J) = 0.0D0
               CALL DAXPY(K,T,A(1,K),1,A(1,J),1)
   80       CONTINUE
   90       CONTINUE
  100    CONTINUE
C
C        FORM INVERSE(U)*INVERSE(L)
C
         NM1 = N - 1
         IF (NM1 .LT. 1) GO TO 140
         DO 130 KB = 1, NM1
            K = N - KB
            KP1 = K + 1
            DO 110 I = KP1, N
               WORK(I) = A(I,K)
               A(I,K) = 0.0D0
  110       CONTINUE
            DO 120 J = KP1, N
               T = WORK(J)
               CALL DAXPY(N,T,A(1,J),1,A(1,K),1)
  120       CONTINUE
            L = IPVT(K)
            IF (L .NE. K) CALL DSWAP(N,A(1,K),1,A(1,L),1)
  130    CONTINUE
  140    CONTINUE
```

```
  150 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE DGEFA(A,N,IPVT,INFO)
C
      INCLUDE 'CONSTMAX.DEF'
C
      INTEGER N,IPVT(1),INFO
      DOUBLE PRECISION A(DIMMAX,1)
C
C     DGEFA FACTORS A DOUBLE PRECISION MATRIX BY GAUSSIAN ELIMINATION.
C
C     DGEFA IS USUALLY CALLED BY DGECO, BUT IT CAN BE CALLED
C     DIRECTLY WITH A SAVING IN TIME IF  RCOND  IS NOT NEEDED.
C     (TIME FOR DGECO) = (1 + 9/N)*(TIME FOR DGEFA) .
C
C     ON ENTRY
C
C        A       DOUBLE PRECISION(DIMMAX, N)
C                THE MATRIX TO BE FACTORED.
C
C        DIMMAX  INTEGER
C                THE LEADING DIMENSION OF THE ARRAY  A .
C
C        N       INTEGER
C                THE ORDER OF THE MATRIX  A .
C
C     ON RETURN
C
C        A       AN UPPER TRIANGULAR MATRIX AND THE MULTIPLIERS
C                WHICH WERE USED TO OBTAIN IT.
C                THE FACTORIZATION CAN BE WRITTEN  A = L*U  WHERE
C                L  IS A PRODUCT OF PERMUTATION AND UNIT LOWER
C                TRIANGULAR MATRICES AND  U  IS UPPER TRIANGULAR.
C
C        IPVT    INTEGER(N)
C                AN INTEGER VECTOR OF PIVOT INDICES.
C
C        INFO    INTEGER
C                = 0  NORMAL VALUE.
C                = K  IF  U(K,K) .EQ. 0.0 .  THIS IS NOT AN ERROR
C                     CONDITION FOR THIS SUBROUTINE, BUT IT DOES
C                     INDICATE THAT DGESL OR DGEDI WILL DIVIDE BY ZERO
C                     IF CALLED.  USE  RCOND  IN DGECO FOR A RELIABLE
C                     INDICATION OF SINGULARITY.
```

```
C
C     LINPACK. THIS VERSION DATED 08/14/78 .
C     CLEVE MOLER, UNIVERSITY OF NEW MEXICO, ARGONNE NATIONAL LAB.
C
C     SUBROUTINES AND FUNCTIONS
C
C     BLAS DAXPY,DSCAL,IDAMAX
C
C     INTERNAL VARIABLES
C
      DOUBLE PRECISION T
      INTEGER IDAMAX,J,K,KP1,L,NM1
C
C
C     GAUSSIAN ELIMINATION WITH PARTIAL PIVOTING
C
      INFO = 0
      NM1 = N - 1
      IF (NM1 .LT. 1) GO TO 70
      DO 60 K = 1, NM1
         KP1 = K + 1
C
C        FIND L = PIVOT INDEX
C
         L = IDAMAX(N-K+1,A(K,K),1) + K - 1
         IPVT(K) = L
C
C        ZERO PIVOT IMPLIES THIS COLUMN ALREADY TRIANGULARIZED
C
         IF (A(L,K) .EQ. 0.0D0) GO TO 40
C
C           INTERCHANGE IF NECESSARY
C
            IF (L .EQ. K) GO TO 10
               T = A(L,K)
               A(L,K) = A(K,K)
               A(K,K) = T
   10       CONTINUE
C
C           COMPUTE MULTIPLIERS
C
            T = -1.0D0/A(K,K)
            CALL DSCAL(N-K,T,A(K+1,K),1)
C
C           ROW ELIMINATION WITH COLUMN INDEXING
C
            DO 30 J = KP1, N
```

```
                  T = A(L,J)
                  IF (L .EQ. K) GO TO 20
                     A(L,J) = A(K,J)
                     A(K,J) = T
   20             CONTINUE
                  CALL DAXPY(N-K,T,A(K+1,K),1,A(K+1,J),1)
   30          CONTINUE
               GO TO 50
   40          CONTINUE
                  INFO = K
   50          CONTINUE
   60       CONTINUE
   70       CONTINUE
            IPVT(N) = N
            IF (A(N,N) .EQ. 0.0D0) INFO = N
            RETURN
            END


            SUBROUTINE  DSCAL(N,DA,DX,INCX)
C
C     SCALES A VECTOR BY A CONSTANT.
C     USES UNROLLED LOOPS FOR INCREMENT EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
            DOUBLE PRECISION DA,DX(1)
            INTEGER I,INCX,M,MP1,N,NINCX
C
            IF(N.LE.0)RETURN
            IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
            NINCX = N*INCX
            DO 10 I = 1,NINCX,INCX
              DX(I) = DA*DX(I)
   10       CONTINUE
            RETURN
C
C        CODE FOR INCREMENT EQUAL TO 1
C
C
C        CLEAN-UP LOOP
C
   20       M = MOD(N,5)
            IF( M .EQ. 0 ) GO TO 40
            DO 30 I = 1,M
```

```
        DX(I) = DA*DX(I)
   30 CONTINUE
      IF( N .LT. 5 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,5
        DX(I) = DA*DX(I)
        DX(I + 1) = DA*DX(I + 1)
        DX(I + 2) = DA*DX(I + 2)
        DX(I + 3) = DA*DX(I + 3)
        DX(I + 4) = DA*DX(I + 4)
   50 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE DAXPY(N,DA,DX,INCX,DY,INCY)
C
C     CONSTANT TIMES A VECTOR PLUS A VECTOR.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL TO ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DA
      INTEGER I,INCX,INCY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF (DA .EQ. 0.0D0) RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C        CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS
C          NOT EQUAL TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DY(IY) = DY(IY) + DA*DX(IX)
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C        CODE FOR BOTH INCREMENTS EQUAL TO 1
C
C
C        CLEAN-UP LOOP
```

```
C
   20 M = MOD(N,4)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DY(I) = DY(I) + DA*DX(I)
   30 CONTINUE
      IF( N .LT. 4 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,4
        DY(I) = DY(I) + DA*DX(I)
        DY(I + 1) = DY(I + 1) + DA*DX(I + 1)
        DY(I + 2) = DY(I + 2) + DA*DX(I + 2)
        DY(I + 3) = DY(I + 3) + DA*DX(I + 3)
   50 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE  DSWAP (N,DX,INCX,DY,INCY)
C
C     INTERCHANGES TWO VECTORS.
C     USES UNROLLED LOOPS FOR INCREMENTS EQUAL ONE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DY(1),DTEMP
      INTEGER I,INCX,INCY,IX,IY,M,MP1,N
C
      IF(N.LE.0)RETURN
      IF(INCX.EQ.1.AND.INCY.EQ.1)GO TO 20
C
C       CODE FOR UNEQUAL INCREMENTS OR EQUAL INCREMENTS NOT EQUAL
C         TO 1
C
      IX = 1
      IY = 1
      IF(INCX.LT.0)IX = (-N+1)*INCX + 1
      IF(INCY.LT.0)IY = (-N+1)*INCY + 1
      DO 10 I = 1,N
        DTEMP = DX(IX)
        DX(IX) = DY(IY)
        DY(IY) = DTEMP
        IX = IX + INCX
        IY = IY + INCY
   10 CONTINUE
      RETURN
C
C       CODE FOR BOTH INCREMENTS EQUAL TO 1
```

```
C
C
C       CLEAN-UP LOOP
C
   20 M = MOD(N,3)
      IF( M .EQ. 0 ) GO TO 40
      DO 30 I = 1,M
        DTEMP = DX(I)
        DX(I) = DY(I)
        DY(I) = DTEMP
   30 CONTINUE
      IF( N .LT. 3 ) RETURN
   40 MP1 = M + 1
      DO 50 I = MP1,N,3
        DTEMP = DX(I)
        DX(I) = DY(I)
        DY(I) = DTEMP
        DTEMP = DX(I + 1)
        DX(I + 1) = DY(I + 1)
        DY(I + 1) = DTEMP
        DTEMP = DX(I + 2)
        DX(I + 2) = DY(I + 2)
        DY(I + 2) = DTEMP
   50 CONTINUE
      RETURN
      END
      INTEGER FUNCTION IDAMAX(N,DX,INCX)
C
C     FINDS THE INDEX OF ELEMENT HAVING MAX. ABSOLUTE VALUE.
C     JACK DONGARRA, LINPACK, 3/11/78.
C
      DOUBLE PRECISION DX(1),DMAX
      INTEGER I,INCX,IX,N
C
      IDAMAX = 0
      IF( N .LT. 1 ) RETURN
      IDAMAX = 1
      IF(N.EQ.1)RETURN
      IF(INCX.EQ.1)GO TO 20
C
C        CODE FOR INCREMENT NOT EQUAL TO 1
C
      IX = 1
      DMAX = DABS(DX(1))
      IX = IX + INCX
      DO 10 I = 2,N
         IF(DABS(DX(IX)).LE.DMAX) GO TO 5
```

```
            IDAMAX = I
            DMAX = DABS(DX(IX))
    5       IX = IX + INCX
   10    CONTINUE
         RETURN
C
C           CODE FOR INCREMENT EQUAL TO 1
C
   20    DMAX = DABS(DX(1))
         DO 30 I = 2,N
            IF(DABS(DX(I)).LE.DMAX) GO TO 30
            IDAMAX = I
            DMAX = DABS(DX(I))
   30    CONTINUE
         RETURN
         END
```

Annexe B

Programme du test des hypervolumes

```
      subroutine testhyp (k,dim,n,r)

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'

      INTEGER k,dim,n
      DOUBLE PRECISION r

      DOUBLE PRECISION v,w

c     partition en k classe
      open (unit=20, file='colonne4.dat', status='old')
c     partition en k-1 classe
      open (unit=30, file='colonne3.dat', status='old')
      call crithyp(k,dim,n,20,w)
      call crithyp(k-1,dim,n,30,v)
      write(*,*)'valeur du crit en', k-1, ' classes',v
      write(*,*)'valeur du crit en', k,' classes',w
      r=w/v
      write(*,*) 'la valeur du quotient est=',r
      end

c     ************************************************
      program test
      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'

      INTEGER k,dim,n
      DOUBLE PRECISION r
```

```
      call testhyp (4,4,80,r)

      end

c     ************************************************

      SUBROUTINE crithyp(k,dim,n,u,w)

c     Crithyp calcule la valeur du critere des hypervolumes, à savoir la somme
c     des mesures de Lebesgues des enveloppes convexes des k classes. Pour
c     calculer cette valeur, on dispose d'une partition en k classes obtenue
c     par sas.

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'


c     Declaration et description des argumemts de crithyp
c     ---------------------------------------------------

      INTEGER k,dim,n,u
      DOUBLE PRECISION w

c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observees sur chaque individus
c     n : Nombre d'individus
c     u : Numero de l'unite associee au fichier contenant les donnees
c     w : Somme des hypervolumes des enveloppes convexes des k classes


c     Declaration et description des variables utilisees dans crithyp
c     ---------------------------------------------------------------

      REAL x(dim,n),c(dimmax,npmax)
      INTEGER cl(n),nb(0:k)
      INTEGER i,j,z,f
      DOUBLE PRECISION volume
      CHARACTER*100 titre
      CHARACTER*8 nom(n)
      integer clnom(n)

c     titre : Contient la premiere ligne du fichier des donnees
c     nom(n) : Vecteur contenant le nom associe a chaque individu (nom(i)
c              contient le nom associe a ieme individu du fichier)
```

```
c     x(dim,n) : Matrice contenant les caracteristiques de tous les individus
c                ( x(j,i) contient la caracteristique j du ieme individu )
c     cl(n) : Vecteur qui contient le numero de la classe des individus
c             ( cl(i) est le numero de la classe du ieme individu )
c     clnom(n) : Vecteur contenant le nom de la classe associe a chaque
c                individu (clnom(i) contient le nom de la classe du ieme
c                individu )
c     nb(0:k) : Vecteur qui contient l'indice du dernier individu de chaque
c               classe ( nb(i) est l'indice du dernier individu dela ieme
c               classe )
c     c(dimmax,npmax) : Matrice qui contiendra les caracteristiques des
c                       individus appartenant a la meme classe
c     z : contient le nombre d'element de la ieme classe a la ieme iteration
c     f : contient la valeue de nb(i-1) a la ieme iteration
c     volume : contient l'hypervolume de la ieme classe a la ieme iteration
c     i,j : indice de boucle


c     Crithyp fait appel a la sous-routine volclasse (k,dim,n,z,f,x,c,volume)
c     qui calcule l'hypervolume de l'enveloppe convexe de chacune des classes.

c     Initialisation de la valeur du critere

      w=0.

c     Lecture du fichier contenant les donnees

c      read(u,100) titre
      do  i=1,n
         read(u,*)nom(i),(x(j,i), j=1,dim),cl(i)
      enddo
c 100  format (a100)



c     Evaluation du vecteur nb

      j=1
      nb(0)=0
      do i=1,k
 200     continue
         if ((cl(j).eq.i).and.(j.le.n))then
            j=j+1
            goto 200
         endif
         nb(i)=j-1
      enddo
```

```
c     Calcul de l'hypervolume de l'enveloppe convexe de chaque classe.

      do i=1,k
         z=nb(i)-nb(i-1)
         f=nb(i-1)
         volume=0.
c         write(*,*) volume
         call volclasse (k,dim,n,z,f,x,c,volume)
c         write (*,*) 'volume de ',i, volume
         w=w+volume
      enddo

      end
c     ******************************************************
      SUBROUTINE volclasse (k,dim,n,z,f,x,c,volume)

c     volclasse calcule l'hypervolume de l'enveloppe convexe de la iéme
c     classe.

      IMPLICIT NONE

      include 'CONSTMAX.DEF'
      include 'FATAL_LIMITS.DEF'

c     Declaration et description des argumemts de volclasse
c     -----------------------------------------------------


      INTEGER k,dim,n,z,f
      DOUBLE PRECISION volume
      real x(dim,n),c(dimmax,z)

c     k : Nombre de classes constituant la partition
c     dim : Nombre de variables observees sur chaque individus
c     n : Nombre d'individus
c     z : contient le nombre d'element de la ieme classe
c     f : contient la valeue de nb(i-1)
c     x(dim,n) : Matrice contenant les caracteristiques de tous les individus
c                ( x(j,i) contient la caracteristique j du ieme individu
c     c(dimmax,npmax) : Matrice qui contient les caracteristiques des
c                       individus appartenant a la meme classe
c     volume : contient l'hypervolume


c     Declaration et description des variables utilisees dans volclasse
c     -----------------------------------------------------------------
```

```
      INTEGER l,j

c     l,j : indice de boucles

c     volclasse fait appel a la sous-routine convexe. La declaration de ses
c     arguments est :

      Real          face    (dimmax,nfmax)
      INTEGER hyper1 (3,nhmax)
      INTEGER face1  (nfmax)
      INTEGER nh,nf,h
      DOUBLE PRECISION surface
      DOUBLE PRECISION hyperplan (dimmax,nhmax)
      DOUBLE PRECISION vectra    (dimmax)


      do l=1,dim
         do j=1,z
            c(l,j)=x(l,f+j)
         enddo
      enddo
c      do l=1,z
c         write(*,*) (c(j,l), j=1,dim)
c      enddo
      volume=0.
c      z=75
c      open (unit=60, file='test2.dat', status='old')
c      z=75
c      do l=1,75
c         read(60,*) (c(j,l), j=1,dim)
c      enddo
c      do l=1,z
c         write(*,*) (c(j,l), j=1,dim)
c      enddo
c      call convexe (c,hyperplan,vectra,face,hyper1,face1,
c     +     dim,z,nh,nf,volume,h,surface,.false.)
c
c      write(*,*) 'volume de test',volume
c      close (60)
      call convexe (c,hyperplan,vectra,face,hyper1,face1,
     +     dim,z,nh,nf,volume,h,surface,.false.)


      end

c La procédure convexe est la même que celle du programme du gaptest
```